PAD-Interface

Programmierhandbuch



Copyright

Herausgeber:

APEX Automationstechnik GmbH

Harxbütteler Straße 3

D-38110 Braunschweig

Telefon 05307 - 9213-0

Telefax 05307 - 9213-30

INHALT

1 Einführung

Über das PDnet (Prozeßdatennetzwerk) werden Daten zwischen verschiedenen Endgeräten (SPS, Workstations, PCs) ausgetauscht. Jedes Endgerät wird über einen PDnet-Controller (PAD) mit dem PDnet verbunden. Mit dem Projektierungsprogramm NetPro werden die PADs im PDnet konfiguriert und die Quellen und Ziele der zu übertragenen Prozeßdaten festgelegt. Die Firmware der PADs überträgt die geänderten Daten zwischen den PADs. Der Datenaustausch zwischen PAD und SPS erfolgt über (Sonder-) Funktionsbausteine, die im SPS-Programm aufgerufen werden. Auf dem PC und auf Workstations wird die PAD-Interface Bibliothek benötigt, um die Daten zwischen dem Anwenderprogramm und dem PAD auszutauschen.

Die vorliegende PAD-Interface Bibliothek bildet die Schnittstelle zwischen der Applikation und dem PDnet. Sie enthält API-Funktionen (Application Programming Interface) zur Überwachung der PDnet-Controller, zum Senden und Empfangen von Telegrammen sowie zur Übertragung von Prozeßvariablen über das virtuelle Datenmodell (VDM) des PDnet.

Auf die PDnet-Controller sollte nur über die API-Funktionen der PAD-Interface Bibliothek zugegriffen werden, da ein direkter Zugriff auf den PAD-Speicher Fehlfunktionen hervorrufen kann. Die PAD-Interface Bibliothek bietet folgende Vorteile:

- Alle Hardwarezugriffe werden Zentral in einer Bibliothek vorgenommen und koordiniert.
- Änderungen, welche das Interface zwischen PAD und PC betreffen, können von der Software "abgefangen" werden; somit ist die Software mit späteren Versionen der Hardware ohne größere Änderung einsetzbar.
- Die Bibliothek benutzt ein einheitliches Hardware-API, welches die Zugriffe auf die Speicherseiten des PADs koordiniert.
- Greifen mehrere Prozesse auf einen PAD zu, synchronisiert die Bibliothek gleichzeitige Zugriffe auf die Datenbereiche des PAD.

Die Bibliothek Überwacht und Behandelt die Änderungen von Flagbytes in den VDM-Datenzellen.

Die PAD-Interface Bibliothek wurde für Systeme entwickelt, welche mit einem PAD ausgestattet sind. Die Bibliothek liegt für die folgenden Systeme vor:

System	Betriebssystem	Compiler
IBM-PC (und 101% kompatible)	DOS (Real Mode)	Borland Pascal 7.0
		Borland C++ 3.1
	DOS (Protected Mode)	Borland Pascal 7.0
	Windows 3.1	Borland Pascal 7.0
		Borland C++ 3.1
	Windows 95	Borland Delphi 2.0
		Borland C++ 4.5
		Microsoft Visual C++ 2.0
		Microsoft Visual C++ 4.0
		Watcom C++ 10.0
	Windows NT 3.5x/4.00	Borland Delphi 2.0
		Borland C++ 4.5
		Microsoft Visual C++ 2.0
		Microsoft Visual C++ 4.0
		Watcom C++ 10.0
	OS/2 Warp 3/4	Borland C++ 2.0 für OS/2
		VisualAge C++ 3.0
		Watcom C++ 10.0
	QNX 3.21 (Protected Mode) (*)	C86
	QNX 4.22 (Protected Mode)	Watcom C 9.5
DEC AlphaStation	OpenVMS/AXP V6.2/7.0	DEC C V5.2
SUN-IPC/IPX	SUN-OS 4.1.2	GNU-C (ANSI-C)
SUN Ultra 5	Solaris 2.6	SunPro C 4.2
Motorola-8420	System V/m88k (*)	GNU-C (ANSI-C)

ANMERKUNG

Die mit (*) gekennzeichneten Versionen der Bibliothek sind in Vorbereitung und noch nicht verfügbar.

Um die Bibliothek verwenden zu können, muß der entsprechende Compiler vorhanden sein. Die Bibliothek übernimmt die gesamte Kommunikation mit dem PAD. Die Bibliothek arbeitet bei den Parametern mit Zeigern und Datenstrukturen, der Umgang damit sollte dem Programmierer geläufig sein.

Dieses Programmierhandbuch der PAD-Interface Bibliothek besteht aus folgenden Teilen:

- Teil 1 „Einleitung“ bietet einen kurzen Überblick über die PAD-Interface Bibliothek.
- Teil 2 „Installation Hardware“ beschreibt welche PDnet-Controller auf der jeweiligen Plattform unterstützt werden.
- Teil 3 „Installation Treiber“ beschreibt für jede Plattform die Installation evt. benötigter Gerätetreiber für die PDnet-Controller.

- Teil 4 „Installation Bibliothek“ befaßt sich mit den Dateien der PAD-Interface Bibliothek und deren Installation.
- Teil 5 „Typen Referenz“ enthält die in der PAD-Interface Bibliothek definierten Typen.
- Teil 6 „Konstanten Referenz“ befaßt sich mit den Konstanten der PAD-Interface Bibliothek.
- Teil 7 „API Referenz“ beschreibt die API Funktionen der PAD-Interface Bibliothek.
- Teil 8 „Beispielprogramm“ erklärt die Compilierung des Beispielprogramms.
- Teil 9 „Programmierung“ beschäftigt sich mit der Programmierung von Treibern für das PDnet.

2 Installation Hardware

Dieses Kapitel beschreibt, welche PDnet-Controller auf der jeweiligen Plattform von der PAD-Interface Bibliothek unterstützt werden.

2.1 IBM-PC DOS (Real Mode)

Von der PAD-Interface Bibliothek für IBM-PC DOS (Real Mode) werden folgende PDnet-Controller unterstützt:

PDnet-Controller	LAN-Anschluß
PAD-PC/K	KOAX
PAD-PC/Kr	KOAX redundant
PAD-PC/L	LWL
PAD-PC/Lr	LWL redundant
PAD-PC/PL	Plastik-LWL
PAD-PC/PLr	Plastik-LWL redundant
PAD-PG/K	KOAX

Den Einbau entnehmen Sie bitte dem Handbuch des PDnet-Controllers.

Die Kommunikation zwischen PAD und PC wird über ein Shared-MemoryInterface abgewickelt. Da der PC im Real Mode jedoch nur 1 MByte adressieren kann, besteht keine Möglichkeit, den gesamten Speicher des PAD linear im PC-Adreßraum einzublenden. Aus diesem Grund greift der PC über ein Speicherfenster auf den PAD zu. Dazu wird der Adreßraum des PAD in mehrere gleich große, fortlaufend angeordnete Seiten unterteilt, welche über ein Adreßregister (I/O-Port) ausgewählt werden. Die Zählung beginnt bei Seite 0.

Der PAD-PC ist in der Regel mit 896 KB RAM und 128 KB Flash-Memory ausgestattet. Er bietet genügend Speicher für größere VDM-Datenmodelle. Das Speicherfenster des PAD-PC ist im PC 64 kByte groß.

Der PAD-PG hat weniger Speicher als der PAD-PC. Er ist als Programmiergerät gedacht, kann aber auch kleine VDM-Datenmodelle verwalten. Das Speicherfenster des PAD-PG ist im PC 4 kByte groß.

Die Größe des Speichers für VDM-Datenmodelle wird für jeden PAD-Typ im Projektierungsprogramm NetPro angezeigt.

ACHTUNG

Damit der PAD-PC bzw. PAD-PG in einem PC läuft, sind die folgenden Bedingungen unbedingt einzuhalten:

- Der auf der Karte gewählte Adreßbereich muß im PC frei sein, und darf durch keine andere Karte belegt sein (Speicher- und I/O-Adreßbereich)
- Es darf kein Cache im Bereich des SharedMemoryInterface aktiviert sein; läßt sich der Cache nicht für einzelne Speicherbereiche deaktivieren, so ist er komplett abzuschalten.
- Wird mit Speichermanagern (EMM386, QEMM, ...) gearbeitet, ist der Speicherbereich des SharedMemoryInterface für die Speicherverwaltung zu sperren (Exclude)
- Wird mit einem Multitasking Betriebssystem gearbeitet (QEMM, OS/2, ...) darf nie mehr als ein Programm gleichzeitig gestartet werden, welches auf den PAD zugreift. Für derartige Anwendungen werden Gerätetreiber benötigt, welche eine Koordinierung der einzelnen Zugriffe durchführen.

2.2 IBM-PC DOS (Protected Mode)

Von der PAD-Interface Bibliothek für IBM-PC DOS (Protected Mode) werden folgende PDnet-Controller unterstützt:

PDnet-Controller	LAN-Anschluß
PAD-PC/K	KOAX
PAD-PC/Kr	KOAX redundant
PAD-PC/L	LWL
PAD-PC/Lr	LWL redundant
PAD-PC/PL	Plastik-LWL
PAD-PC/PLr	Plastik-LWL redundant
PAD-PG/K	KOAX

Den Einbau entnehmen Sie bitte dem Handbuch des PDnet-Controllers.

2.3 IBM-PC Windows 3.1

Von der PAD-Interface Bibliothek für IBM-PC Windows 3.1 werden folgende PDnet-Controller unterstützt:

PDnet-Controller	LAN-Anschluß
PAD-PC/K	KOAX
PAD-PC/Kr	KOAX redundant
PAD-PC/L	LWL
PAD-PC/Lr	LWL redundant
PAD-PC/PL	Plastik-LWL
PAD-PC/PLr	Plastik-LWL redundant
PAD-PG/K	KOAX

Den Einbau entnehmen Sie bitte dem Handbuch des PDnet-Controllers.

2.4 IBM-PC Windows 95

Von der PAD-Interface Bibliothek für IBM-PC Windows 95 werden folgende PDnet-Controller unterstützt:

PDnet-Controller	LAN-Anschluß
PAD-PC/K	KOAX
PAD-PC/Kr	KOAX redundant
PAD-PC/L	LWL
PAD-PC/Lr	LWL redundant
PAD-PC/PL	Plastik-LWL
PAD-PC/PLr	Plastik-LWL redundant
PAD-PG/K	KOAX

Den Einbau entnehmen Sie bitte dem Handbuch des PDnet-Controllers.

2.5 IBM-PC Windows NT 3.5x/4.00

Von der PAD-Interface Bibliothek für IBM-PC Windows NT 3.5x/4.00werden folgende PDnet-Controller unterstützt:

PDnet-Controller	LAN-Anschluß
PAD-PC/K	KOAX
PAD-PC/Kr	KOAX redundant
PAD-PC/L	LWL
PAD-PC/Lr	LWL redundant
PAD-PC/PL	Plastik-LWL
PAD-PC/PLr	Plastik-LWL redundant
PAD-PG/K	KOAX
PAD-PCI/K	KOAX
PAD-PCI/Kr	KOAX redundant
PAD-PCI/L	LWL
PAD-PCI/Lr	LWL redundant

Den Einbau entnehmen Sie bitte dem Handbuch des PDnet-Controllers.

2.6 IBM-PC OS/2 Warp 3/4

Von der PAD-Interface Bibliothek für IBM-PC OS/2 Warp werden folgende PDnet-Controller unterstützt:

PDnet-Controller	LAN-Anschluß
PAD-PC/K	KOAX
PAD-PC/Kr	KOAX redundant
PAD-PC/L	LWL
PAD-PC/Lr	LWL redundant
PAD-PC/PL	Plastik-LWL
PAD-PC/PLr	Plastik-LWL redundant
PAD-PG/K	KOAX

Den Einbau entnehmen Sie bitte dem Handbuch des PDnet-Controllers.

2.7 IBM-PC QNX 3.21 (Protected Mode)

Von der PAD-Interface Bibliothek für IBM-PC QNX 3.21 (Protected Mode) werden folgende PDnet-Controller unterstützt:

PDnet-Controller	LAN-Anschluß
PAD-PC/K	KOAX
PAD-PC/Kr	KOAX redundant
PAD-PC/L	LWL
PAD-PC/Lr	LWL redundant
PAD-PC/PL	Plastik-LWL
PAD-PC/PLr	Plastik-LWL redundant
PAD-PG/K	KOAX

Den Einbau entnehmen Sie bitte dem Handbuch des PDnet-Controllers.

2.8 IBM-PC QNX 4.22 (Protected Mode)

Von der PAD-Interface Bibliothek für IBM-PC QNX 4.22 (Protected Mode) werden folgende PDnet-Controller unterstützt:

PDnet-Controller	LAN-Anschluß
PAD-PC/K	KOAX
PAD-PC/Kr	KOAX redundant
PAD-PC/L	LWL
PAD-PC/Lr	LWL redundant
PAD-PC/PL	Plastik-LWL
PAD-PC/PLr	Plastik-LWL redundant
PAD-PG/K	KOAX

Den Einbau entnehmen Sie bitte dem Handbuch des PDnet-Controllers.

2.9 DEC AlphaStation OpenVMS/AXP 6.2/7.0

Von der PAD-Interface Bibliothek für DEC AlphaStation OpenVMS/AXP 6.2/7.0 werden folgende PDnet-Controller unterstützt:

PDnet-Controller	LAN-Anschluß
PAD-PC/L	LWL
PAD-PC/Lr	LWL redundant
PAD-PC/PL	Plastik-LWL
PAD-PC/PLr	Plastik-LWL redundant

Der PDnet-Controller kann in AlphaStation-System mit ISA-Bus oder in AlphaServer-Systemen mit EISA-Bus eingebaut werden.

Vor dem Ausschalten des Systems ist auf jeden Fall ein Shutdown des Systems auszuführen!

Die Konfiguration der Hardwareparameter erfolgt über DIP-Schalter auf der Karte. Die DIP-Schalter sind im Benutzerhandbuch des PDnet-Controllers beschrieben.

Für den PDnet-Controller sind folgende Hardwareparameter zu konfigurieren:

1. IRQ Interrupt

   Für den JDDriver ist „kein Interrupt" zu konfigurieren.

2. I/O-Portadresse

   Für den JDDriver ist eine der möglichen Basisadressen für den I/O-Port auszuwählen und einzustellen. Die I/O-Basisadresse darf von keinem anderem EISA-/ISA-Gerät im System genutzt werden. Auf Konsolebene des Systems kann dies mit Hilfe des Kommandos *ISACFG* oder dem ECU (EISA Configuration Utility) überprüft werden. Über die I/O-Basisadresse werden die I/O-Ports der Karte adressiert. Die I/O-Basisadresse muß für jede E/ISA-Karte im System eindeutig sein. Ab der eingestellten Adressen werden 8 I/O-Ports (8 Byte) genutzt.

3. Speicheradresse

   Für den JDDriver ist eine der möglichen Speicheradressen auszuwählen und einzustellen. Ab dieser Startadresse wird ein 64 kByte großes Speicherfenster im Adreßraum des EISA-/ISA-Bus eingeblendet. Dieser Adreßraum darf nicht von anderen EISA-/ISA-Geräten im System genutzt werden. Auf Konsolebene des Systems kann dies mit Hilfe des Kommandos *ISACFG* oder dem ECU (EISA Configuration Utility) überprüft werden. Die Speicheradresse muß eindeutig für alle EISA-/ISA-Geräte im System sein. Die belegten Adreßräume dürfen sich nicht überlappen.

Nach der Konfiguration der Hardwareparameter kann die Karte entsprechend den Hinweisen in der Hardwaredokumentation der AlphaStation und der Hardwaredokumentation des PAD-PC eingebaut werden.

2.10 SUN IPC/IPX (SUN-OS 4.1.2)

Von der PAD-Interface Bibliothek für SUN IPC/IPX (SUN-OS 4.1.2) werden folgende PDnet-Controller unterstützt:

PDnet-Controller	LAN-Anschluß
PAD-SBus/K	KOAX
PAD-SBus/Kr	KOAX redundant

Den Einbau entnehmen Sie bitte dem Handbuch des PDnet-Controllers.

2.11 SUN Ultra 5 (Solaris 2.6)

Von der PAD-Interface Bibliothek für SUN Ultra 5 (Solaris 2.6) werden folgende PDnet-Controller unterstützt:

PDnet-Controller	LAN-Anschluß
PAD-PCI/K	KOAX
PAD-PCI/Kr	KOAX redundant
PAD-PCI/L	LWL
PAD-PCI/Lr	LWL redundant

Den Einbau entnehmen Sie bitte dem Handbuch des PDnet-Controllers.

2.12 Motorola-8420 (System V/m88k)

Von der PAD-Interface Bibliothek für Motorola-8420 (System V/m88k)werden folgende PDnet-Controller unterstützt:

PDnet-Controller	LAN-Anschluß
PAD-VME/K	KOAX
PAD-VME/Kr	KOAX redundant
PAD-VME/L	LWL
PAD-VME/Lr	LWL redundant

Den Einbau entnehmen Sie bitte dem Handbuch des PDnet-Controllers.

3 Installation Treiber

Dieses Kapitel beschreibt, auf welchen Betriebsystemen zusätzliche Treiber installiert werden müssen, die den Zugriff auf die PDnet-Controller Hardware steuern.

3.1 IBM-PC DOS (Real Mode)

Die PAD-Interface Bibliothek für IBM-PC DOS (Real Mode) greift direkt auf den PDnet-Controller zu und benötigt deshalb keine zusätzlichen Hardwaretreiber.

3.2 IBM-PC DOS (Protected Mode)

Die PAD-Interface Bibliothek für IBM-PC DOS (Real Mode) greift über die Servicefunktion der Version 0.9 des DPMI-Servers auf den PDnet-Controller zu und benötigt deshalb keine zusätzlichen Hardwaretreiber.

ACHTUNG

Weitere Angaben zum PAD-PC und PAD-PG finden Sie im Kapitel IBM-PC DOS (Real Mode)

3.3 IBM-PC Windows 3.1

Die PAD-Interface Bibliothek für IBM-PC Windows 3.1 greift über die Servicefunktion der Version 0.9 des DPMI-Servers auf den PDnet-Controller zu und benötigt deshalb keine zusätzlichen Hardwaretreiber. Deshalb ist die Bibliothek nur im Protected Mode lauffähig.

Weitere Angaben zum PAD-PC und PAD-PG finden Sie im Kapitel IBM-PC DOS (Real Mode).

3.4 IBM-PC Windows 95

Da beim Betriebssystem Windows 95 nicht direkt auf die Hardware zugegriffen werden kann, erfolgt der Zugriff auf den PDnet-Controller (PAD) über einen Gerätetreiber. Dazu muß nach dem Einbau des PDnet-Controllers der Gerätetreiber PADITF32.VXD in das Verzeichnis SYSTEM von Windows 95 kopiert werden und der Gerätetreiber und die vom PDnet-Controller (PAD) verwendeten Ressourcen (Speicherbereich und E/A-Bereich) in die Registrierdatenbank von Windows 95 eingetragen werden.

Folgende Dateien befinden sich im Root-Verzeichnis auf der mitgelieferten Diskette:

Dateiname	Beschreibung	Weitergabe
README.TXT	Änderungen und Ergänzungen der gedruckten Dokumentation	NEIN
README95.TXT	Anleitung zur Installation des Gerätetreibers unter Windows 95	JA
PADITF32.INF	Informationsdatei für die Installation des Gerätetreibers unter Windows 95	JA
PADITF32.VXD	Windows 95 Gerätetreiber für PDnet-Controller im PC	JA

Weitere Angaben zum PAD-PC und PAD-PG finden Sie im Kapitel IBM-PC DOS (Real Mode). Hinweise zur Installation des Gerätetreibers finden Sie unten bzw. in der Datei README95.TXT.

Falls Sie auf ihrem System den Gerätetreiber APEX-PAD.VXD installiert haben, sollten Sie diesen deinstallieren, bevor Sie den Gerätetreiber PADITF32.VXD installieren, da auf einen PAD nur ein Gerätetreiber zugreifen kann.

Der Gerätetreiber PADITF32.VXD unterstützt mehrere lokale PADs in einem PC. Damit diese unterschieden werden können, erhält jeder hinzugefügte PAD eine Gerätenummer, über die das Anwenderprogramm auf den PAD zugreift. Die Gerätenummer wird im Geräte-Manager von Windows 95 vor jedem eingetragenen PDnet-Controller in eckigen Klammern angezeigt. Nach der Installation des Gerätetreibers PADITF32.VXD muß Windows 95 neu gestartet werden, um die Gerätenummer im Geräte-Manager anzuzeigen. Falls Ihr Anwenderprogramm nur einen PAD unterstützt, können Sie die Gerätenummer evt. vernachlässigen.

3.4.1 Gerätetreiber installieren

Um den Gerätetreiber PADITF32.VXD unter Windows 95 zu installieren, benötigen Sie eine Diskette in deren Root-Verzeichnis sich die Dateien PADITF32.VXD (Gerätetreiber), PADITF32.INF (Informationsdatei) und README95.TXT (Installationsanleitung) befinden. Damit Ihr späterer Kunde den Gerätetreiber unter Windows 95 installieren kann, sollten Sie die Dateien README95.TXT, PADITF32.INF und PADITF32.VXD in das Root-Verzeichnis der Lieferdiskette Ihres Produktes kopieren.

Der Gerätetreiber für den PDnet-Controller wird folgendermaßen installiert:

- Legen Sie die Diskette mit den Dateien PADITF32.VXD und PADITF32.INF in das Diskettenlaufwerk.
- Wählen Sie im Startmenü "Einstellungen".
- Öffnen Sie die Programmgruppe "Systemsteuerung".
- Doppelklicken Sie auf das Symbol "Hardware" der Systemsteuerung und klicken Sie auf "Weiter".
- Antworten Sie auf die Frage, ob Windows 95 neue Hardware suchen soll, mit "Nein" und klicken Sie auf "Weiter".

Falls Sie den Gerätetreiber PADITF32.VXD schon einmal auf Ihrem PC installiert hatten, befindet sich in der Liste der Hardwaretypen die Geräteklasse "APEX PDnet-Controller".

- Wählen Sie den Hardwaretyp "APEX PDnet-Controller" aus und klicken Sie auf "Weiter".

Wenn Sie den Gerätetreiber PADITF32.VXD zum ersten Mal auf Ihrem PC installieren, ist der Hardwaretyp "APEX PDnet-Controller" noch nicht in der Liste vorhanden.

- Klicken Sie in der Liste der Hardwaretypen auf die Geräteklasse "Andere Komponenten" und auf "Weiter".
- Klicken Sie auf die Schaltflächen "Diskette" und "OK".

Anschließend:

- Wählen Sie das verwendete Model "APEX PDnet-Controller PAD-PC" oder "APEX PDnet-Controller PAD-PG" und klicken Sie auf "Weiter".

Nach dem Kopieren des Gerätetreibers schlägt Windows 95 freie Ressourcen für den PDnet-Controller vor.

Falls ein Ressourcen Konflikt auftritt, müssen Sie den PAD konfigurieren (Gerätetreiber konfigurieren).

- Klicken Sie auf "Weiter" und auf "Weiter".
- Entfernen Sie die Diskette aus dem Laufwerk.

Wenn die Ressourcen nicht mit den eingestellten Adressen auf dem PDnet-Controller übereinstimmen:

- Beantworten Sie die Frage zum Herunterfahren des Computers mit "Nein".
- Ändern Sie die Konfiguration (siehe: Gerätetreiber konfigurieren).
- Fahren Sie den Computer herunter.

Wollen Sie die vorgeschlagenen Ressourcen verwenden:

- Beantworten Sie die Frage zum Herunterfahren des Computers mit "Ja".
- Schalten Sie nach dem Herunterfahren von Windows 95 den Computer aus.
- Überprüfen Sie, ob die Ressourcen mit den eingestellten Adressen auf dem PDnet-Controller übereinstimmen.
- Ändern Sie gegebenenfalls die Stellung der DIP-Schalter auf dem PDnet-Controller.

Nach dem Neustart von Windows 95 benutzt der Gerätetreiber PADITF32.VXD (APEX PDnet-Controller Driver) die im Geräte-Manager eingestellten Ressourcen.

3.4.2 Gerätetreiber konfigurieren

Wenn Sie den Gerätetreiber PADITF32.VXD bereits installiert haben und die eingestellte Konfiguration (Speicherbereich und E/A-Bereich) ändern wollen:

- Wählen Sie im Startmenü "Einstellungen".
- Öffnen Sie die Programmgruppe "Systemsteuerung".
- Doppelklicken Sie auf das Symbol "System" der Systemsteuerung
- Wählen Sie die Registerkarte "Geräte-Manager" im Register System.
- Doppelklicken Sie auf den Gerätetyp "APEX PDnet-Controller".
- Selektieren Sie den zu konfigurierenden "APEX PDnet-Controller PAD-PC" bzw. "APEX PDnet-Controller PAD-PG" und klicken Sie auf die Schaltfläche "Eigenschaften".
- Klicken Sie auf die Registerkarte "Ressourcen" im Eigenschaftsfenster.

Wenn keine Ressourcen angezeigt werden:

- Klicken Sie auf die Schaltfläche "Manuell konfigurieren".

Anschließend:

- Wählen Sie in der Liste "Ressourceneinstellungen" die zu ändernde Ressource aus.
- Klicken Sie auf die Schaltfläche "Einstellung ändern".
- Wählen Sie eine Einstellung aus, die nicht mit anderen Geräten in Konflikt liegt, und klicken Sie auf "OK".

Nach der Konfiguration der Ressourcen:

- Klicken Sie auf die Registerkarte "Allgemein" im Eigenschaftsfenster.
- Aktivieren Sie das Kontrollkästchen "Ausgangskonfiguration" in der Gerätenutzung, wenn es noch nicht gesetzt ist.
- Schließen Sie das Eigenschaftsfenster mit "OK".
- Schließen Sie den Geräte-Manager mit der Schaltfläche "Schließen".

Wenn das Dialogfeld "Geänderte Systemeinstellungen" erscheint:

- Klicken Sie auf die Schaltfläche "Ja", um Windows 95 zu beenden.

Wenn die gewählten Ressourcen nicht mit den eingestellten Adressen auf dem PDnet-Controller übereinstimmen:

- Schalten Sie nach dem herunterfahren von Windows 95 den Computer aus.
- Stellen Sie die DIP-Schalter des PDnet-Controllers auf die Adressen der verwendeten Ressourcen.
- Starten Sie Windows 95 neu.

3.4.3 Gerätetreiber deinstallieren

Wenn Sie den Gerätetreiber PADITF32.VXD entfernen wollen:

- Wählen Sie im Startmenü "Einstellungen".
- Öffnen Sie die Programmgruppe "Systemsteuerung".
- Doppelklicken Sie auf das Symbol "Software" der Systemsteuerung.
- Selektieren Sie den zu entfernenden "APEX PDnet-Controller PAD-PC".
- Klicken Sie auf die Schaltfläche "Hinzufügen/Entfernen".
- Schließen Sie das Softwarefenster mit "OK".

3.5 IBM-PC Windows NT 3.5x/4.00

Da beim Betriebssystem Windows NT nicht direkt auf die Hardware zugegriffen werden kann, erfolgt der Zugriff auf den PDnet-Controller (PAD) über einen Gerätetreiber. Dazu muß nach dem Einbau des PDnet-Controllers der Gerätetreiber PADITF32.SYS in das Verzeichnis SYSTEM32\DEVICE und die Systemsteuerungs-applikation PADITFNT.CPL in das Verzeichnis SYSTEM von Windows NT kopiert werden. Anschließend kann in der Systemsteuerung über das Symbol „PDnet“ der Gerätetreiber und die vom PDnet-Controller (PAD) verwendeten Ressourcen (Speicherbereich und I/O Port Anschluß) in die Registrierdatenbank von Windows NT eingetragen werden.

Folgende Dateien befinden sich im Root-Verzeichnis auf der mitgelieferten Diskette:

Dateiname	Beschreibung	Weitergabe
README.TXT	Änderungen und Ergänzungen der gedruckten Dokumentation	NEIN
READNT4.TXT	Anleitung zur Installation des Gerätetreibers unter Windows NT 4.00	JA
PADITFNT.CPL	Systemsteuerungsapplikation für die Installation des Gerätetreibers unter Windows NT	JA
PADITF32.SYS	Windows NT Gerätetreiber für PDnet-Controller im PC	JA

Weitere Angaben zum PAD-PC und PAD-PG finden Sie im Kapitel IBM-PC DOS (Real Mode). Hinweise zur Installation des Gerätetreibers finden Sie unten bzw. in der READNT4.TXT

ACHTUNG

Damit Sie unter Windows NT Gerätetreiber installieren können, müssen Sie sich zunächst als ein Benutzer mit Administatorrechten anmelden. Nur mit diesen Rechten können Gerätetreiber in das Windows NT Verzeichnis SYSTEM32\DEVICE kopiert werden und die Registrierdatenbank von Windows NT verändert werden!

Falls Sie auf ihrem System den Gerätetreiber APEX-PAD.SYS installiert haben, sollten Sie diesen deinstallieren, bevor Sie den Gerätetreiber PADITF32.SYS installieren, da auf einen PAD nur ein Gerätetreiber zugreifen kann.

Der Gerätetreiber PADITF32.SYS unterstützt mehrere lokale PADs in einem PC. Damit diese unterschieden werden können, erhält jeder hinzugefügte PAD eine Gerätenummer, über die das Anwenderprogramm auf den PAD zugreift. Die Gerätenummer wird im Register "Netzwerk" in der Registerkarte "Netzwerkkarte" vor jedem eingetragenen PDnet-Controller in eckigen Klammern angezeigt. Falls Ihr Anwenderprogramm nur einen PAD unterstützt, können Sie die Gerätenummer evt. vernachlässigen.

3.5.1 Gerätetreiber installieren

Zur Installation des PDnet-Gerätetreibers für Windows NT müssen Sie die Dateien auf die Festplatte kopieren:

- Wenn Sie keine Administrator Rechte haben, melden Sie sich als Administrator an.
- Legen Sie das Installationsmedium in das Laufwerk ein.
- Öffnen Sie über das Windows Startmenü ein Eingabeaufforderungsfenster oder den Windows Explorer.
- Kopieren Sie folgende Dateien auf die Festplatte:

Datei	Ziel	Beschreibung
PADITF32.SYS	<WINDIR>\SYSTEM32 \DRIVERS	PDnet-Gerätetreiber
PADITFNT.CPL	<WINDIR>\SYSTEM32	Systemsteuerungsregister für PDnet-Gerätetreiber

Anschließend:

- Konfigurieren Sie die PDnet-Controller in der Systemsteuerung, wie unten beschrieben.

Bei jedem Start von Windows NT wird nun der Gerätetreiber PADITF32.SYS (APEX PDnet-Controller Driver) automatisch gestartet. Ein Anwendungsprogramm kann über den Gerätetreiber auf den PDnet-Controller (PAD) zugreifen.

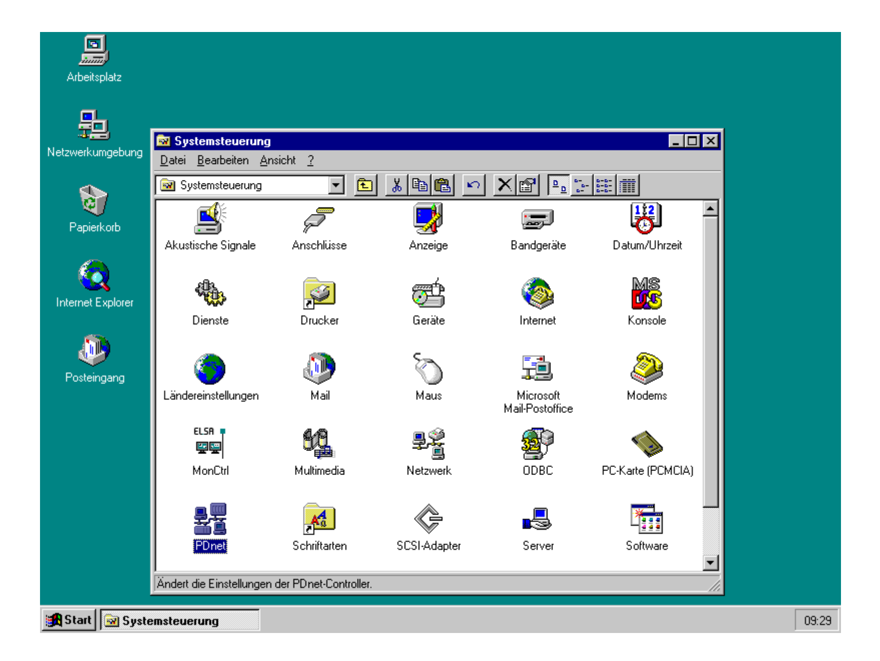
3.5.2 Gerätetreiber konfigurieren

Nachdem Sie den PDnet-Gerätetreiber `PADITF32.SYS` installiert haben müssen Sie die installierten PDnet-Controller konfigurieren:

- Wählen Sie im Startmenü `Einstellungen`.
- Öffnen Sie die Programmgruppe `Systemsteuerung`.
- Doppelklicken Sie auf das Symbol `PDnet` der `Systemsteuerung`.

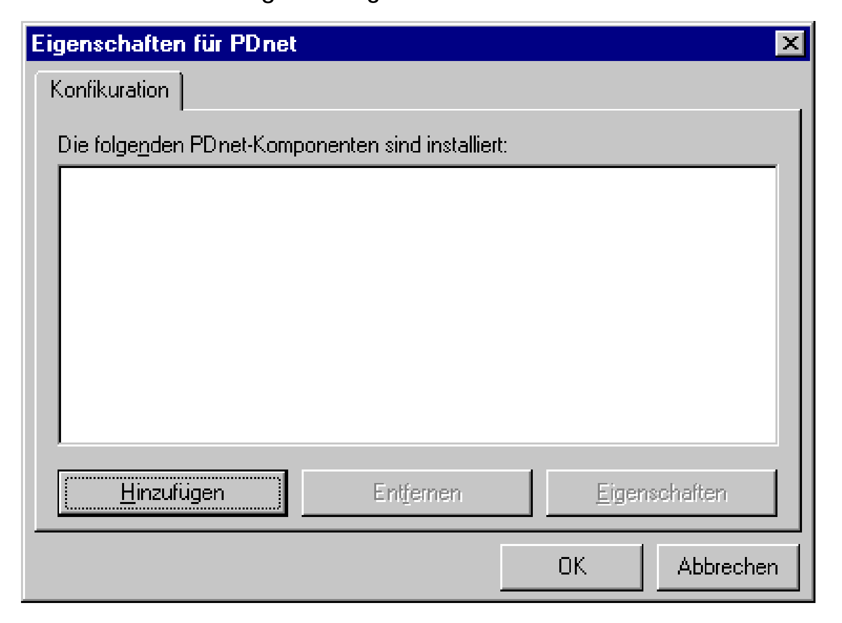
3 Installation Treiber
3.5 IBM-PC **Windows NT 3.5x/4.00**

Bild: Systemsteuerung



Es erscheint das Register `Eigenschaften für PDnet`.

Bild: Eigenschaften für PDnet



In diesem Register können Sie PDnet-Controller hinzufügen, entfernen oder ihre Ressourcen einstellen.

PDnet-Controller hinzufügen

Um einen PDnet-Controller zu installieren:

- Klicken Sie auf die Schaltfläche `Hinzufügen`.

Es erscheint das Dialogfeld `Hardwarekomponente hinzufügen`.

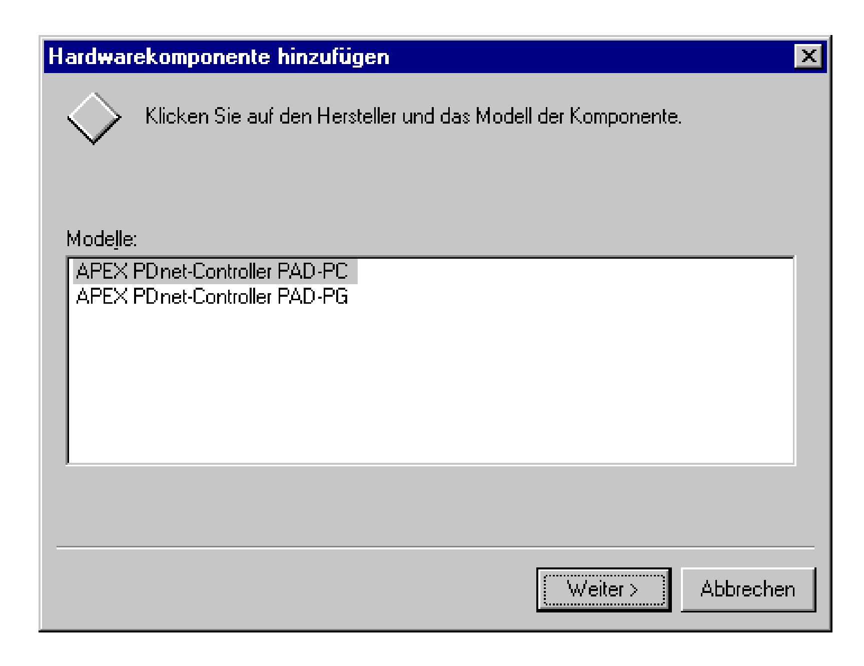


Bild: Hardwarekomponente hinzufügen

- Selektieren Sie den zu installierenden `APEX PDnet-Controller`.
- Klicken Sie auf die Schaltfläche `Weiter`.

Es erscheint das Dialogfeld `Eigenschaften für PDnet-Controller`.

- Ändern Sie die eingestellten Ressourcen, bis sie mit denen des PDnet-Controllers übereinstimmen.
- Klicken Sie auf die Schaltfläche `OK`.

Danach sollte der hinzugefügte `APEX PDnet-Controller` in der Liste `PDnet-Komponenten` erscheinen.

Vor jedem PDnet-Controller steht in eckigen Klammern seine Gerätenummer, über die auf ihn zugegriffen werden kann. Sie benötigen diese Gerätenummer für die Konfigurationsdatei des PDnet-Clients.

Wenn Sie mehrere PDnet-Controller in einem PC installieren wollen:

- Führen Sie für jeden zu installierenden PAD die in oben aufgeführten Installationsschritte durch.

Bei jedem Start von Windows NT wird nun der Gerätetreiber `PADITF32.SYS` (APEX PDnet-Controller Driver) automatisch gestartet.

PDnet-Controller Ressourcen einstellen

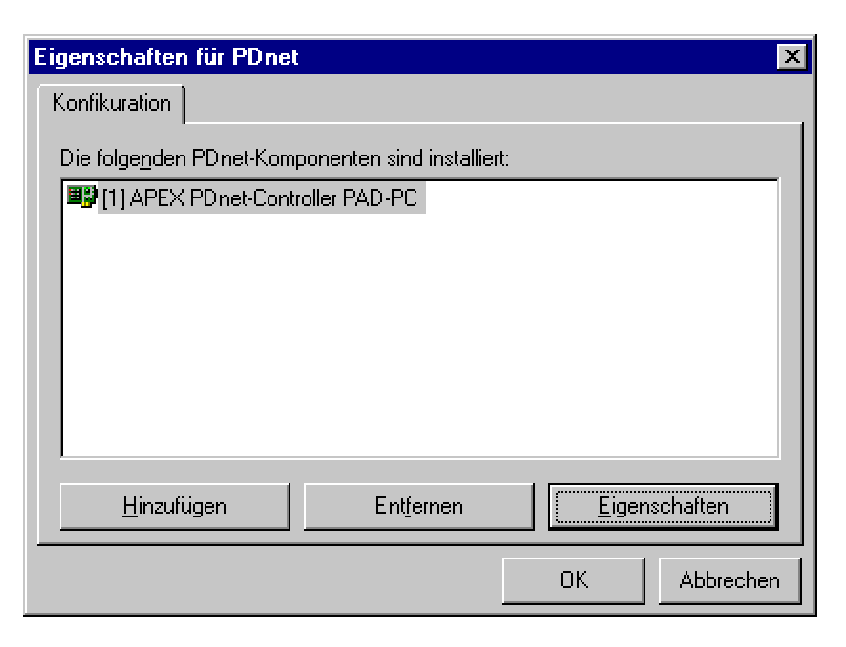
Wenn Sie die Ressourcen eines PDnet-Controllers ändern wollen:

- Selektieren Sie den zu konfigurierenden

3 Installation Treiber
3.5 IBM-PC **Windows NT 3.5x/4.00**

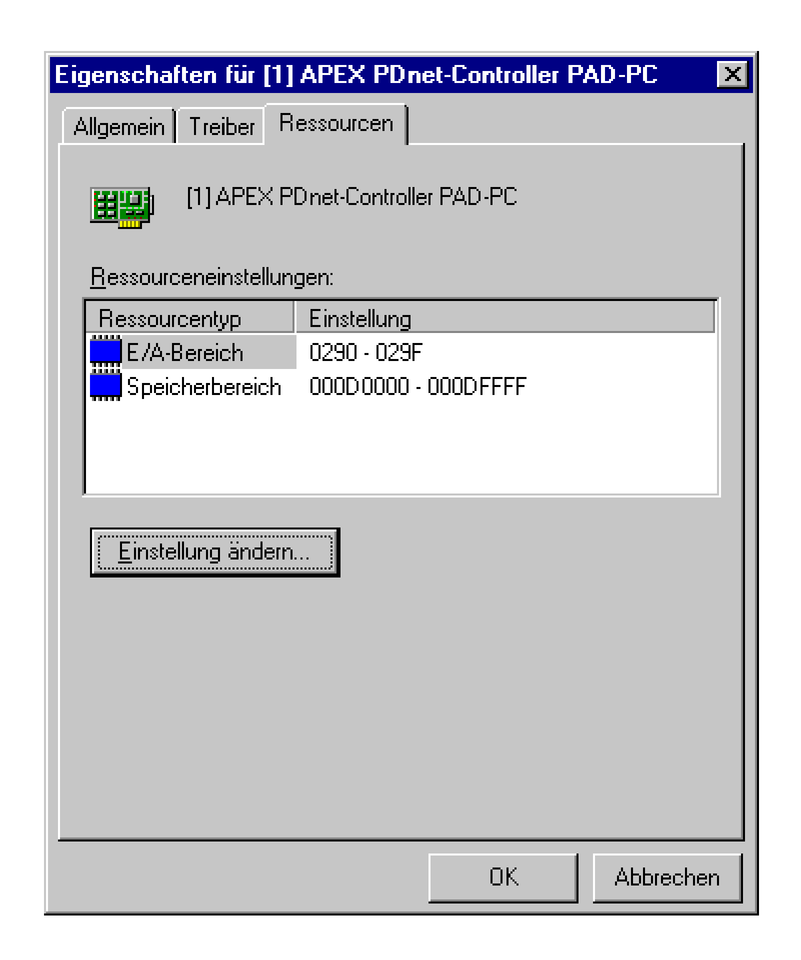
`APEX PDnet-Controller PAD-PC` bzw.
`APEX PDnet-Controller PAD-PG` bzw.
`APEX PDnet-Controller PAD-PCI`

Bild: PDnet-Controller Eigenschaften



- Klicken Sie dann auf `Eigenschaften`.
- Klicken Sie auf die Registerkarte `Ressourcen` im Eigenschaftsfenster.

Bild: Ressourcen



Anschließend:

- Wählen Sie in der Liste `Ressourceneinstellungen` die zu ändernde Ressourcen aus.
- Klicken Sie auf die Schaltfläche `Einstellung ändern`.
- Wählen Sie eine Einstellung aus, die nicht mit anderen Geräten in Konflikt liegt, und klicken Sie auf `OK`.

Nach der Konfiguration der Ressourcen:

- Schließen Sie das Eigenschaftsfenster mit `OK`.
- Schließen Sie das Register `Eigenschaften für PDnet` mit der Schaltfläche `OK`.
- Beenden Sie Windows NT.

Wenn die gewählten Ressourcen nicht mit den eingestellten Adressen auf dem PDnet-Controller übereinstimmen:

- Schalten Sie den Computer aus.
- Stellen Sie die DIP-Schalter des PDnet-Controllers auf die Adressen der verwendeten Ressourcen.
- Starten Sie Windows NT neu.

PDnet-Controller entfernen

Wenn Sie einen installierten PDnet-Controller entfernen wollen:

- Selektieren Sie den zu entfernenden

`APEX PDnet-Controller PAD-PC` bzw.
`APEX PDnet-Controller PAD-PG` bzw.
`APEX PDnet-Controller PAD-PCI`

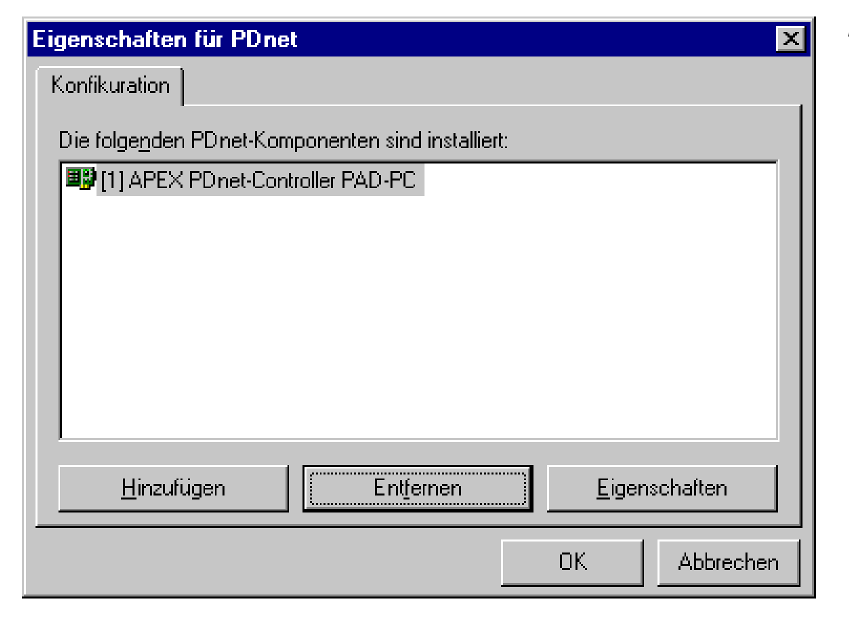


Bild: PDnet-Controller Entfernen

- Klicken Sie dann auf `Entfernen`.

Es erscheint eine Sicherheitsabfrage.

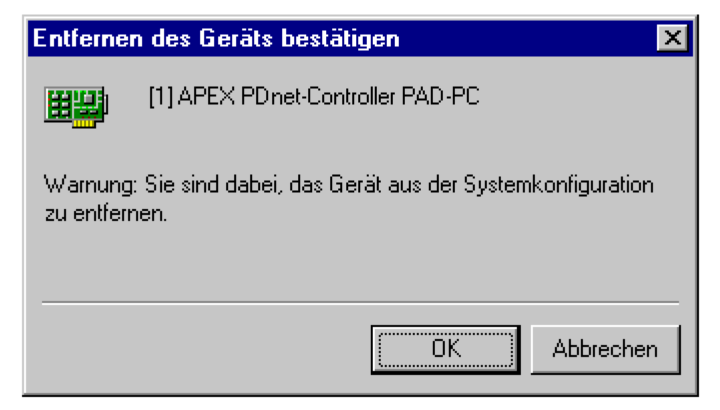


Bild: Entfernen bestätigen

- Klicken Sie auf `OK`.
- Schließen Sie das Register `Eigenschaften für PDnet` mit der Schaltfläche `OK`.
- Beenden Sie Windows NT.

HINWEIS

Wenn Sie alle PDnet-Controller entfernen, wird beim nächsten Start von Windows NT der PDnet-Gerätetreiber nicht mehr geladen.

3.5.3 Gerätetreiber überprüfen

Wenn Sie überprüfen wollen, ob Windows NT die installierten Gerätetreiber korrekt startet:

- Wählen Sie im Startmenü `Einstellungen`.
- Öffnen Sie die Programmgruppe `Systemsteuerung`.
- Doppelklicken Sie auf das Symbol `Geräte` in der Systemsteuerung.

Es erscheint daß Dialogfeld `Geräte`.

- Suchen Sie die Zeile:

`APEX PDnet-Controller Driver`

  – Wenn Sie die Zeile nicht finden, haben Sie den Gerätetreiber für den PDnet-Controller noch nicht installiert oder noch keinen PDnet-Controller in der Systemsteuerung eingetragen.

- Überprüfen Sie, ob hinter dem Gerät der Text steht:

`Gestartet Automatisch`

  – Wenn nicht, liegt ein Konflikt mit anderen Hardware Ressourcen vor. Ändern Sie dann die Ressourcen des PDnet-Controllers oder der anderen Hardware.

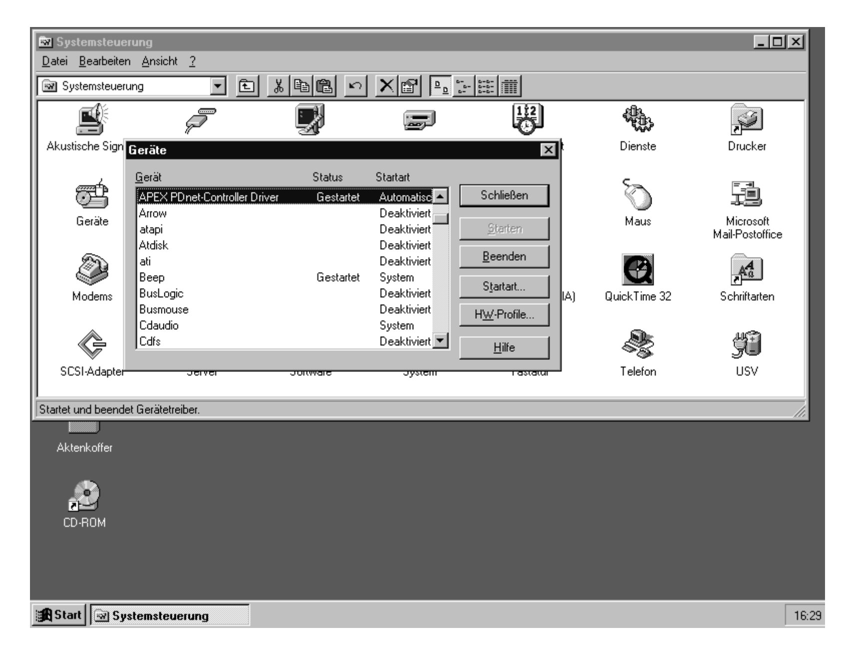


Bild: Gerät APEX PDnet-Controller

- Beenden Sie das Dialogfeld `Geräte` mit dem Schalter `Schließen`

3.5.4 Gerätetreiber Ressourcen überprüfen

Wenn Sie Ressourcenkonflikte anderer Geräte mit dem PDnet-Controller überprüfen wollen:

- Wählen Sie im Startmenü `Programme`.
- Wählen Sie das Menü `Verwaltung (Allgemein)`.
- Klicken Sie auf die `Windows NT-Diagnose`.
- Wählen Sie die Registerkarte `Ressourcen` im Register `Windows NT Diagnose`.
- Klicken Sie auf die Schaltfläche `Geräte`.

Falls in der Liste `Geräte` die Zeile `PDnet` nicht erscheint, wurde der Gerätetreiber `PADITF32.SYS` von Windows NT nicht gestartet:

- Installieren Sie den PDnet-Gerätetreiber PADITF32.SYS oder starten Sie ihn manuell (siehe oben).

Andernfalls:

- Selektieren Sie die Zeile `PDnet` in der Liste `Geräte`.
- Klicken Sie auf die Schaltfläche `Eigenschaften`.

Im Dialog `Eigenschaften von PDnet` werden die vom Gerätetreiber verwendeten Ressourcen angezeigt:

Als I/O-Bereich sollten die Portadressen der installierten PDnet-Controller erscheinen.

Als Speicherbereich sollten die Speicheradressen der installierten PDnet-Controller angezeigt werden. Der Speicher hat bei einem PAD-PC eine Länge von 0x10000 und beim PAD-PG eine Länge von 0x02000.

Falls im Dialogfeld `PDnet` kein I/O-Bereich oder Speicherbereich angezeigt wird, liegt ein Ressourcenkonflikt mit einem anderen Gerät vor:

- Überprüfen Sie dann in den Listen `I/O-Port`, `Speicher` und `Geräte`, ob andere Geräte die gleichen Adressen verwenden und ändern Sie gegebenenfalls die doppelt belegten Adressen dieses Gerätes bzw. die Adressen des PDnet-Controllers.
- Schließen Sie die Dialogfelder durch klicken der Schaltflächen `OK`.

3.5.5 Gerätetreiber deinstallieren

Wenn Sie einen installierten PDnet-Controller entfernen wollen:

- Wählen Sie im Startmenü "Einstellungen".
- Öffnen Sie die Programmgruppe "Systemsteuerung".
- Doppelklicken Sie auf das Symbol "PDnet" in der Systemsteuerung.

Es erscheint das Register "Eigenschaften für PDnet".

- Selektieren Sie in der Liste den zu entfernenden "APEX PDnet-Controller".
- Klicken Sie auf die Schaltfläche "Entfernen".
- Bestätigen Sie das Entfernen des PDnet-Controllers durch Klicken der Schaltfläche "Ja".
- Schließen Sie das Register "Eigenschaften für PDnet" mit der Schaltfläche "OK".
- Starten die den Computer neu.
- Schließen Sie die Systemsteuerung.
- Anschließend können Sie die Dateien löschen:
  `<WINDIR>\SYSTEM32\DRIVERS\PADITF32.SYS`
  `<WINDIR>\SYSTEM32\PADITFNT.CPL`

3.6 IBM-AT OS/2 Warp 3/4

Da beim Betriebssystem OS/2 nicht direkt auf die Hardware zugegriffen werden kann, erfolgt der Zugriff auf den PDnet-Controller (PAD) über einen Einheitentreiber. Dazu muß nach dem Einbau des PDnet-Controllers der Einheitentreiber PADITF32.SYS auf die Festplatte von OS/2 kopiert werden und der Einheitentreiber mit den verwendeten Ressourcen (I/O-Port-Bereich und Speicherbereich) in die Datei CONFIG.SYS eingetragen werden.

Folgende Dateien befinden sich im Root-Verzeichnis auf der mitgelieferten Diskette:

Dateiname	Beschreibung	Weitergabe
README.TXT	Änderungen und Ergänzungen der gedruckten Dokumentation	NEIN
READMEO2.TXT	Anleitung zur Installation des Einheitentreibers unter OS/2	JA
APEX-PAD.SYS	OS/2 Einheitentreiber für PDnet-Controller im PC	JA

Weitere Angaben zum PAD-PC und PAD-PG finden Sie im Kapitel IBM-PC DOS (Real Mode). Hinweise zur Installation des Einheitentreibers finden Sie unten bzw. in der Datei READMEO2.TXT.

3.6.1 Einheitentreiber installieren

Damit Ihr späterer Kunde den Einheitentreiber unter OS/2 installieren kann, sollten Sie die Dateien READMEO2.TXT und APEX-PAD.SYS in das Root-Verzeichnis der Lieferdiskette Ihres Produktes kopieren.

Der Einheitentreiber für den PDnet-Controller wird wie folgt installiert:

- Legen Sie die Diskette mit der Datei APEX-PAD.SYS in das Diskettenlaufwerk.
- Öffnen Sie die Programmgruppe "System".
- Doppelklicken Sie auf das Symbol "Befehlszeilen" der System Symbolanzeige.
- Doppelklicken Sie auf das Symbol "OS/2-Fenster" der Befehlszeilen Symbolanzeige.
- Kopieren Sie im OS/2-Fenster Sie den Treiber von der Diskette auf die Festplatte mit dem Befehl:

```
COPY A:\APEX-PAD.SYS C:\OS2\BOOT /V
```


Nach dem Kopieren des Einheitentreibers muß dieser in der Datei CONFIG.SYS eingetragen werden.

- Geben Sie im OS/2-Fenster den Befehl ein:

```
E C:\CONFIG.SYS
```

- Bewegen Sie im Editor (E.EXE) den Cursor an das Textende und erweitern Sie die Datei CONFIG.SYS um die Zeile:

```
DEVICE=C:\OS2\BOOT\APEX-PAD.SYS 290 D0000
```

Durch die eingefügte Zeile wird der Einheitentreiber APEX-PAD.SYS beim nächsten Systemstart geladen.

Nach dem Dateinamen APEX-PAD.SYS werden als Parameter in hexadezimaler Schreibweise die I/O-Portadresse und danach die Startadresse des Speicherfensters des eingebauten PDnet-Controllers angegeben. Diese Adressen müssen mit den DIP-Schaltern auf dem PDnet-Controller übereinstimmen und dürfen durch keine anderen Hardwarekomponenten belegt sein.

- Ändern Sie Zahlen hinter APEX-PAD.SYS, falls Sie andere Adressen auf dem PDnet-Controller eingestellt haben.
- Speichern Sie die Datei CONFIG.SYS mit dem Menüpunkt "Datei/Sichern".
- Beenden Sie den Editor durch Drücken der Tasten "Alt-F4".
- Schließen Sie das OS/2-Fenster mit dem Befehl:

```
EXIT
```

- Klicken Sie mit der rechten Maustaste auf eine freie Stelle der Arbeitsoberfläche und wählen Sie den Menüpunkt "Systemab-schluß".
- Beantworten Sie die Frage zum Durchführen eines Systemab-schlusses mit "OK".
- Schalten Sie nach dem Herunterfahren von OS/2 den Computer aus und überprüfen Sie, ob die Ressourcen mit den eingestell-ten Adressen auf dem PDnet-Controller übereinstimmen.
- Ändern Sie gegebenenfalls die DIP-Schalter auf dem PDnet-Controller.

Nach dem Neustart von OS/2 benutzt der Einheitentreiber APEX-PAD.SYS (APEX PDnet-Controller Driver) die in der Datei CON-FIG.SYS eingestellten Ressourcen. Ein Anwendungsprogramm kann dann über den Einheitentreiber auf den PDnet-Controller (PAD) zugreifen.

Falls Sie auch DOS-Programme unter OS/2 ausführen wollen, sollten Sie das PAD-Speicherfenster vor EMS-Zugriffen schützen:

- Öffnen Sie für die DOS-Fenster Symbole mit der rechten Maustaste den Menüpunkt "Einstellungen".

- Betätigen Sie auf der Seite "Sitzung" den Schalter "DOS-Einstellungen".
- Lassen Sie sich alle Einstellungen anzeigen und wählen Sie in der Liste "MEM_EXCLUDE_REGIONS" aus.
- Tragen Sie hier den Speicherbereich des PAD-Speicherfensters ein, z. B.:

```
D0000-E0000
```

3.6.2 Einheitentreiber konfigurieren

Falls Sie den Einheitentreiber APEX-PAD.SYS bereits installiert haben und die eingestellte Konfiguration (I/O-Port-Bereich und Speicherbereich) ändern wollen, gehen Sie wie folgt vor:

- Öffnen Sie ein "OS/2-Fenster".
- Geben Sie im OS/2-Fenster den Befehl ein:

```
E C:\CONFIG.SYS
```

- Suchen Sie im Editor (E.EXE) in der Datei CONFIG.SYS die Zeile in welcher der Einheitentreiber APEX-PAD.SYS auftaucht. Z. B.:

```
DEVICE=C:\OS2\BOOT\APEX-PAD.SYS 290 D0000
```

- Ändern Sie hinter dem Treiber APEX-PAD.SYS den I/O-Port-Bereich und den folgenden Speicherbereich. Beide Angaben sind hexadezimal.
- Achten Sie darauf, das der Speicherbereich 5-stellig ist. Die Adressen müssen mit den DIP-Schaltern auf dem PDnet-Controller übereinstimmen und dürfen durch keine anderen Hardwarekomponenten belegt sein.
- Speichern Sie die Datei CONFIG.SYS mit dem Menüpunkt "Datei/Sichern".
- Beenden Sie den Editor durch Drücken der Tasten "Alt-F4".
- Schließen Sie das OS/2-Fenster mit dem Befehl:

```
EXIT
```

- Führen Sie einen Systemabschluß durch.

Wenn die gewählten Ressourcen nicht mit den eingestellten Adressen auf dem PDnet-Controller übereinstimmen:

- Schalten Sie den Computer aus und stellen Sie die DIP-Schalter auf gleichen Adressen, wie die gewählten Ressourcen.
- Starten Sie OS/2 neu.
- Überprüfen Sie, ob die Geräteeinstellungen korrekt sind.

3.6.3 Einheitentreiber deinstallieren

Wenn Sie den Einheitentreiber APEX-PAD.SYS entfernen wollen, gehen Sie wie folgt vor:

- Öffnen Sie ein "OS/2-Fenster".
- Geben Sie im OS/2-Fenster den Befehl ein:

```
E C:\CONFIG.SYS
```

- Suchen Sie im Editor (E.EXE) in der Datei CONFIG.SYS die Zeile in welcher der Einheitentreiber APEX-PAD.SYS auftaucht. Z. B.:

```
DEVICE=C:\OS2\BOOT\APEX-PAD.SYS 290 D0000
```

- Löschen Sie diese Zeile aus der Datei CONFIG.SYS.
- Speichern Sie die Datei CONFIG.SYS mit dem Menüpunkt "Datei/Sichern".
- Beenden Sie den Editor durch Drücken der Tasten "Alt-F4".
- Schließen Sie das OS/2-Fenster mit dem Befehl:

```
EXIT
```

- Führen Sie einen Systemabschluß durch.

3.7 IBM-PC QNX 3.21 (Protected Mode)

Die PAD-Interface Bibliothek für IBM-PC QNX 3.21 (Protected Mode) greift direkt auf den PDnet-Controller zu und benötigt deshalb keine zusätzlichen Hardwaretreiber.

3.8 IBM-PC QNX 4.22 (Protected Mode)

Die PAD-Interface Bibliothek für IBM-PC QNX 4.22 (Protected Mode) greift direkt auf den PDnet-Controller zu und benötigt deshalb keine zusätzlichen Hardwaretreiber.

3.9 DEC AlphaStation OpenVMS/AXP 6.2/7.0

Da beim Betriebssystem OpenVMS nicht direkt auf die Hardware zugegriffen werden kann, erfolgt der Zugriff auf den PDnet-Controller (PAD) über einen Gerätetreiber. Dazu muß nach dem Einbau des PDnet-Controllers der Gerätetreiber PDNET$JDDRIVER.EXE auf die Festplatte kopiert werden und die verwendeten Ressourcen (I/O-Port-Bereich und Speicherbereich) eingetragen werden.

Folgende Dateien befinden sich im Archiv PDNET010.A auf der mitgelieferten Diskette:

Dateiname	Beschreibung	Weitergabe
SYS$LOADABLE_IMAGES: PDNET$JDDRIVER.EXE	OpenVMS/AXP Gerätetreiber JDDriver für PDnet-Controller	JA
SYS$MANAGER: PDNET$CONFIGURE.COM	Kommandoprozedur zur Konfiguration der Kommandos zum Laden des JDDrivers	JA
SYS$TEST:PDNET$IVP.EXE	Programmimage der IVP (Installation Verification Procedure)	JA
SYS$HELP: PDNET010.RELEASE_NOTES	Textdatei mit den Release Notes V1.0	JA

Weitere Angaben zum PAD-PC finden Sie im Kapitel IBM-PC DOS (Real Mode). Hinweise zur Installation des Gerätetreibers finden Sie unten.

Weitere Informationen zur Installation und Nutzung des PDnet/AXP - OpenVMS/AXP Gerätetreiber für PAD-PC, über andere Hardware und über Gerätetreiber finden sich in folgenden Dokumentationen:

1. Hardwaredokumentation des Alpha-Systems, z.B. *Digital AlphaStation 255 User Information*
2. Dokumente, welche die Hardwarekonfiguration für Alpha-Systeme beschreiben, z.B. *OpenVMS AXP Version 7.1 Release Notes and Installation Procedures*
3. Systemmanagement-Dokumentation, *OpenVMS System Manager's Manual*, *OpenVMS System Management Utilities Reference Manual*
4. Programmierer-Dokumentation, *Introduction to OpenVMS System Services*
5. *OpenVMS System Service Reference Manual*
6. Hardwaredokumentation, *PAD-PC Benutzerhandbuch*

3.9.1 ISA-Bus Hardwarekonfiguration mit ISACFG

Vor der Installation des Gerätetreibers müssen die vom PDnet-Controller verwendeten Ressourcen dem Betriebssystem bekannt gemacht werden.

Für AlphaStations, die den ISA-Bus unterstützen, gibt es verschiedene Möglichkeiten, die Hardwareparameter zu konfigurieren:

1. ISACFG Utility

   Auf der Konsolebene kann das ISACFG Utility benutzt werden, um die im NVRAM des Systems abgelegte Einstellung der Hardwareparameter für ISA-Karten, die ISA Configuration Table, zu verwalten. Mit dem ISACFG Utility können neue Daten eingetragen, vorhandene Daten modifiziert und gelöscht werden. In der Tabelle sind die Hardwareparameter zu jedem ISA-Gerät abgelegt. Beim Booten des Systems wird die ISA Configuration Table gelesen und die Hardwareressourcen werden vergeben und die Hardware wird initialisiert. Die ISA Configuration Table enthält keine Informationen über die Konfiguration der den ISA-Geräten zugeordneten OpenVMS Devices. In die ISA Configuration Table müssen auch nicht alle Hardwareparameter eingetragen werden.

2. ISA Configuration File

   Beim Booten des Systems wird das ISA Configuration File SYS$SPECIFIC:[SYSMGR]ISA_CONFIG.DAT gesucht und gelesen. Die Datei enthält Informationen über die Hardwareparameter und die OpenVMS Devices, für die ein Gerätetreiber zu laden ist. Dabei wird zum Teil auf Informationen aus der ISA Configuration Table zurückgegriffen.

3. Autoconfiguration Table

   Ab OpenVMS Version 7.1 gibt es die Möglichkeit, analog zur ISA Configuration Table, die beim Systemstart zu konfigurierenden Geräte in einer Datendatei zu beschreiben. Auch hier wird aber auf die ISA Configuration Table zurückgegriffen.

Bei der Verwendung des ISA Configuration File oder der Autoconfiguration Table kann ein automatisches Laden der erforderlichen Gerätetreiber beim Systemstart und eine Konfiguration der (OpenVMS) Geräte erfolgen. Alternativ können die Gerätetreiber manuell geladen werden.

In jedem Fall ist ein Eintrag in der ISA Configuration Table mit dem ISACFG Utility zu erzeugen. Für den JDDriver des PDnet/AXP wird das manuelle Laden des Gerätetreibers verwendet, da nur wenig ISA-Bus Hardwareparameter zu konfigurieren sind. Das ISACFG Utility ist in der Hardwaredokumentation des Alpha-Systems komplett beschrieben.

Hardwareparameter in die ISA Configuration Table eintragen

Die Einstellung der Hardwareparameter für ISA-Karten werden wie folgt in der ISA Configuration Table eingetragen:

- Gehen Sie auf die Konsolebene des Systems (Prompt >>>). Sie erreichen die Konsolebene durch drücken des Resettasters beim Einschalten der Workstation oder durch Eingabe des DCL Befehls shutdown im DECterm Fenster.
- Zeigen Sie die aktuelle ISA Configuration Table an mit dem Befehl:

```
>>> show config
```

Die ausgegebene Tabelle sieht z.B. so aus:

```
ISA
Slot Device Name  Type       Enabled BaseAddr IRQ
0
     0      MOUSE Embedded   Yes     60       12
     1      KBD   Embedded   Yes     60       1
     2      COM1  Embedded   Yes     3f8      4
```

Die ISA Configuration Table besteht aus eine Folge von Einträgen. Jeder Eintrag wird durch eine Steckplatznummer (Slot) und eine Gerätenummer (Device) identifiziert und enthält die Daten zu jeweils einem ISA-Gerät. Der PDnet-Controller PAD-PC besteht nur aus einem Gerät. Deshalb ist die Gerätenummer immer 0.

Die ISA-Steckplätze einer AlphaStation haben keine festen Steckplatznummern (Slot). Es gibt keine vorgegebene Zuordnung zwischen den Steckplatznummern in der ISA Configuration Table (genutzt in den ISACFG Kommandos) und einer Numerierung der Steckplätze im System. Für die Steckplatznummern (Slot) kann eine Zahl größer als 0 verwendet werden. Die Steckplatznummern sollten von 1 beginnend fortlaufend verwendet werden. Dabei sollte man eine Systematik verwenden, welche eine Zuordnung der Steckplatznummern (Slot) zu den Steckplätzen des Systems ermöglicht. Z.B. kann Slot 1 der unterste oder linke Steckplatz sein.

Mit dem ISACFG Utility ist ein neuer Eintrag in der ISA Configuration Table zu erzeugen und mit den entsprechenden Werten zu belegen. Ist der Eintrag für den anzulegenden Slot bereits vorhanden, ist dieser vorher zu löschen oder ein anderer Slot zu wählen.

- Erzeugen Sie einen neuen Eintrag in der ISA Configuration Table an mit dem Befehl:

```
>>> isacfg -slot <slot#> -dev 0 -mk -etyp 1
           -enadev 1 -totdev 1
           -iobase0 <iobase#>
           -membase0 <membase#>
           [-memlen0 <memlen#>]
           -handle “PDnet“
```

Dabei müssen folgende Werte eingesetzt werden:

1. slot#
   Die Steckplatznummer des ausgewählten Slots.

2. iobase#
   Die hexadezimale I/O-Portadresse, z.B. 290.

3. membase#
   Die hexadezimale Startadresse des Speicherfensters als absolute Speicheradresse. Die absolute Speicheradresse zur Segmentadresse x000 ist x0000, also z.B. zu Segment D000 ist die absolute Speicheradresse D0000.

4. memlen#
   Die hexadezimale Größe des Speicherfensters ist optional. Sie erfolgt in Bytes (also 10000 für 64 kByte).

Beispiel für einen PAD-PC auf Slot 1 mit I/O-Basisadresse 290 und Speicheradresse D0000:

```
>>> isacfg -slot 1 -dev 0 -mk -etyp 1
    -enadev 1 -totdev 1 -iobase0 290
    -membase0 D0000 -memlen0 10000
    -handle “PDnet“
```

- Aktivieren Sie die Änderungen der ISA Configuration Table mit dem Befehl:

```
>>> init
```

- Zeigen Sie die aktuelle ISA Configuration Table an mit dem Befehl:

```
>>> show config
```

In der ISA Configuration Table sollte nun auf dem gewählten Slot der Name PDnet und die gewählte I/O-Basisadresse angezeigt werden:

```
ISA
Slot Device Name  Type       Enabled BaseAddr IRQ
0
     0      MOUSE Embedded   Yes     60       12
     1      KBD   Embedded   Yes     60       1
     2      COM1  Embedded   Yes     3f8      4
1
     0      PDnet Singleport Yes    290
```

- Sie können alle eingestellten Adressen der Slots anzeigen mit dem Befehl:

```
>>> isacfg -all
```

In der angezeigten Konfigurationstabelle werden alle Geräte und Slots angezeigt. Für den jeweiligen Bus (ISA oder EISA) muß die konfigurierte Karte sichtbar sein.

- Booten Sie nach der Überprüfung der Konfiguration das System mit dem Befehl:

```
>>> boot
```

- Achten Sie beim Booten des Systems auf Fehlermeldungen. Zu Fehlern kann es kommen, wenn z.B. im ISA Configuration File ein Gerät konfiguriert wird, welches auf die gleichen Ressourcen zugreift wie der eingestellte PDnet-Controller.

<u>Hardwareparameter aus der ISA Configuration Table entfernen</u>

Die Einstellung der Hardwareparameter für ISA-Karten werden wie folgt aus der ISA Configuration Table entfernt:

- Gehen Sie auf die Konsolebene des Systems (Prompt >>>). Sie erreichen die Konsolebene durch drücken des Resettasters beim Einschalten der Workstation oder durch Eingabe des DCL Befehls shutdown im DECterm Fenster.
- Zeigen Sie die aktuelle ISA Configuration Table an mit dem Befehl:

```
>>> show config
```

- Suchen Sie den zu entfernenden Slot und geben Sie den Befehl ein:

```
>>> isacfg -slot <slot#> -dev 0 -rm
```

  Beispiel für einen PAD-PC auf Slot 1:

```
>>> isacfg -slot 1 -dev 0 -rm
```

- Aktivieren Sie die Änderungen der ISA Configuration Table mit dem Befehl:

```
>>> init
```

3.9.2 EISA-Bus Hardwarekonfiguration mit ECU

Das ECU ist auf einer gesonderten Diskette, die mit dem System geliefert wird, enthalten. ECU ist ein menügesteuertes Programm mit ausführlicher Hilfefunktion. Weitere Informationen sind in der Hardwaredokumentation des Systems zu finden.

Mit Hilfe des ECU können folgende Aufgaben realisiert werden:

1. Ermitteln der aktuellen EISA-Konfiguration. Die beinhaltet die Übersicht über die genutzten und freien Ressourcen (IRQs, I/O-Portadressen, ...) und die Belegung der einzelnen Slots.
2. Karten zum System hinzufügen oder entfernen.
3. Anzeige und Modifikation der Daten zu den einzelnen Slots.

4. Anzeige der entsprechend den ausgewählten Hardwareparametern notwendigen DIP-Schalter Stellungen (nur, wenn Konfigurationsdatei zu der jeweiligen Karte vorhanden ist).
5. Abspeichern der EISA-Konfiguration.

Zu jeder E/ISA-Karte gehört eine EISA-Konfigurationsdatei. Diese Datei beschreibt die Karte, insbesondere die DIP-Schalter Stellungen, die zu den einzelnen Einstellungen der Hardwareparameter gehören. Ist keine Konfigurationsdatei vorhanden, kann eine ISA-Karte konfiguriert werden, indem die Konfigurationsdatei einer allgemeinen ISA-Karte (generic ISA board) genutzt wird. Damit ist es möglich, eine beliebige ISA-Karte im System zu installieren und die von der Karte genutzten Ressourcen zu vergeben und zu konfigurieren.

In einem EISA-Bus System haben alle Steckplätze feste Steckplatznummern. Die Numerierung ist der Hardwaredokumentation des Systems zu entnehmen. Auf die Steckplätze (Slots) wird im ECU über die Steckplatznummer bezug genommen. Im ECU erhält man auch eine Übersicht über die Steckplatznummern und die Belegung der einzelnen Steckplätze.

- Rufen Sie das Programm ECU auf mit dem Befehl:

```
>>> ecu
```

- Wählen Sie im ECU einen freien EISA-Bus-Steckplatz (Slot).
- Tragen Sie die Ressourcen des PDnet-Controllers ein.

Ist für die Karte kein EISA-Konfigurationsdatei vorhanden, dann muß ein Eintrag auf der Basis einer Standard ISA-Karte erfolgen.

Als Ressourcen sind die eingestellte I/O-Basisadresse und die Basisadresse des Speicherfensters zu vergeben. Die Größe des Speicherfensters kann eingegeben werden. Als IRQ oder DMA-Kanal ist 0 bzw. nicht genutzt einzutragen.

- Beenden Sie das Programm ECU.
- Booten Sie nach der Überprüfung der Konfiguration das System mit dem Befehl:

```
>>> boot
```

- Achten Sie beim Booten des Systems auf Fehlermeldungen. Zu Fehlern kann es kommen, wenn z.B. im ISA Configuration File ein Gerät konfiguriert wird, welches auf die gleichen Ressourcen zugreift wie der eingestellte PDnet-Controller.

3.9.3 Gerätetreiber installieren

Die Software Installation erfolgt unter OpenVMS mit Hilfe des Systemprogrammes VMSINSTAL. Das Programm ist in der Softwaredokumentation von OpenVMS näher beschrieben (siehe *OpenVMS System Managers Utilities Reference Manual*).

Betriebssystem

PDnet/AXP kann unter OpenVMS/AXP Version 6.1 bis Version 7.1 installiert werden. Zusätzliche Informationen über die unterstützten Betriebssystemversionen sind in den Release Notes nachzulesen.

Bei einem Wechsel der Betriebssystemversion ist PDnet/AXP erneut zu installieren. Der im System installierte Gerätetreiber (JDDriver) kann mit der neuen Betriebssystemversion nicht weiter genutzt werden. Aus dem Softwarekit kann durch die Installation ein neuer Gerätetreiber erzeugt werden, sofern die Betriebssystemversion unterstützt wird.

Zeit

Die Installation benötigt ca. 3 - 15 Minuten. Dies hängt vom verwendeten Datenträger ab, auf dem der Softwarekit geliefert wird.

Privilegien

Zur Durchführung der Installation mit VMSINSTAL unter OpenVMS sind Prozeßprivilegien erforderlich. Der Account muß das SETPRV Privileg oder mindestens die Privilegien

SYSPRV, CMKRNL, WORLD

haben. Um die SYSMAN Kommandos zu laden des Gerätetreibers auszuführen, muß der Account unter dem das Laden ausgeführt wird das Privileg SETPRV oder mindestens die Privilegien

OPER, SYSLCK, CMKRNL

haben.

Plattenplatz

Während der Installation werden auf der Systemplatte ca. 5000 freie Blöcke benötigt. Der installierte Kit benötigt ca. 500 freie Blöcke auf der Systemplatte.

Um den freien Plattenplatz auf der Systemplatte zu ermitteln, kann das DCL Kommando

```
$ SHOW DEVICE SYS$SYSDEVICE
```

verwendet werden. Man erhält eine Ausgabe, die dem folgenden Beispiel entspricht:

```
Device       Device  Error Volume      Free    Trans Mnt
Name         Status  Count Label       Blocks  Count Cnt
EQA014$DKA0 Mounted     0 EQUICONSYS  1118534   111   1
```

In der Spalte „Free Blocks“ sind die auf der Systemplatte vorhandenen freien Blöcke angegeben.

VMSINSTAL Anforderungen

Wenn VMSINSTAL aufgerufen wird, werden folgende Anforderungen überprüft:

- das aktuelle Gerät und Verzeichnis sollte SYS$UPDATE sein
- der Account muß hinreichende Privilegien haben
- der Account muß hinreichende Quotas und Limits haben
- es sollten keine anderen Benutzer im System angemeldet sein

VMSINSTAL benötigt mindestens die folgenden Quotas:

```
ASTLM = 24, BIOLM = 18, BYTLM = 18000,
DIOLM = 18, ENQLM = 30, FILLM = 40.
```

Falls VMSINSTAL ein Problem mit den Mindestanforderungen erkennt, erfolgt eine entsprechende Mitteilung und die Abfrage, ob die Installation fortgesetzt werden soll. Unter Umständen kann die Installation durch Eingabe von YES fortgesetzt werden. Um die Installation abzubrechen, ist NO einzugeben. Nach der Behebung des Problems kann VMSINSTAL erneut aufgerufen werden.

Backup der Systemplatte

Beim Beginn der Installation fragt VMSINSTAL ab, ob ein Backup der Systemplatte gemacht wurde. Es ist üblich und dringend angeraten, vor der Installation von PDnet/AXP oder eines anderen Produktes immer eine Sicherung der Systemplatte vorzunehmen.

Zur Durchführung des Backup sind die spezifischen Verfahren für die jeweilige Konfiguration des Systems anzuwenden. Weitere Informationen finden sich in der Softwaredokumentation von OpenVMS/AXP, insbesondere *OpenVMS System Managers Manual* und *OpenVMS System Managers Utilities Reference Manual* (Beschreibung des BACKUP Utility).

OpenVMS Device Namen

Wenn ein Gerätetreiber logisch mit einem PAD-PC verbunden wird, wird ein OpenVMS Gerät (Device Unit) erzeugt. Jedes dieser Geräte hat einen eindeutigen Namen, der zur Identifikation des Gerätes dient. Die Namen unterliegen unter OpenVMS einer einheitlichen Konvention.

Die allgemeine Form der Gerätenamen für PDnet/AXP (Device Units) ist:

JDx0:

- **JD** steht für die zweibuchstabige OpenVMS Mnemonik für Gerätetreiber von Drittfirmen
- **x** ist ein Buchstaben der eindeutig einem PDnet-Controller im System zugeordnet ist
- **0** ist die Gerätenummer (Device Unit), welche bei PDnet-Controllern immer 0 ist.

Die Controller-Buchstaben können für die PDnet-Controller frei vergeben werden. Die Namen der Geräte müssen im System eindeutig sein. Der erste PDnet-Controller bekommt üblicherweise

den Namen JDA0: und das zweite JDB0:. Dabei gibt es keine vorgegebene Zuordnung zwischen EISA-/ISA-Steckplatznummern und dem Controller-Buchstaben. Es ist aber eine sehr gute Idee, zur Steckplatznummer *x* den *x*-ten Buchstaben des Alphabetes als Controller-Buchstaben zu verwenden. Damit hat man eine direkte Beziehung zwischen den OpenVMS Gerätenamen und den PDnet-Controllern.

Durchführung der Softwareinstallation

Der Gerätetreiber für den PDnet-Controller wird wie folgt installiert:

- Legen Sie die Diskette mit der Datei PDNET010.A in das Diskettenlaufwerk.
- Starten Sie das Betriebsystem OpenVMS.
- Melden Sie sich mit System Account an (Benutzer SYSTEM).
- Öffnen Sie ein DECterm Fenster.
- Rufen Sie VMSINSTAL auf. Als Parameter werden der Archivname und der Device Name des Diskettenlaufwerk angegeben:

```
$ @sys$update:vmsinstal pdnet010 dva0:
```

  Die Installation kann jederzeit mit "Ctrl+Y" abgebrochen werden.

- Beantworten Sie die von der Installation gestellten Fragen. Die Vorgaben in eckigen Klammern [] können mit der Return Taste übernommen werden.
- Logen Sie sich nach der Installation aus, da VMSINSTAL die Symboltabellen löscht und neu überschreibt:

```
$logout
```

Nach der Installation des Gerätetreibers muß dieser konfiguriert und geladen werden.

3.9.4 Kommando zum Laden des Gerätetreibers

Der Gerätetreiber wird mit *SYSMAN IO CONNECT* Kommandos manuell geladen. *SYSMAN* ist ein OpenVMS System Utility. Es ist im *OpenVMS System Management Utilities Reference Manual* beschrieben.

Die Kommandoprozedur *PDNET$CONFIGURE* erzeugt eine Kommandoprozedur mit den *SYSMAN IO CONNECT* Kommandos zum Laden des Gerätetreibers. Durch Aufruf der erzeugten Kommandoprozedur wird der Gerätetreiber geladen.

Zum Erzeugen der Kommandos zum Laden des Gerätetreibers:

- Öffnen Sie ein DECterm Fenster.
- Geben Sie den Befehl ein:

```
$ @SYS$MANAGER:PDNET$CONFIGURE
```

Die Kommandoprozedur arbeitet interaktiv. Die Konfiguration erfolgt in mehreren Schritten:

- Geben Sie den Namen der Kommandoprozedur zum Laden des Gerätetreibers ein. Der Name ist standardmäßig *SYS$STARTUP:PDNET$STARTUP.COM*. Gibt es die angegebene Kommandoprozedur bereits, dann kann man diese Überschreiben, Erweitern oder einen anderen Namen auswählen.
- Geben Sie den Slots des PDnet-Controllers ein, für den der Gerätetreiber zu laden ist. Die Angabe des Slots entspricht der Slot-Nummer, die in ISACFG oder ECU für den Slot verwendet wurde.
- Geben Sie für den PDnet-Controller die eingestellten I/O-Basisadresse und Speicheradresse ein. Diese Werte werden in der Kommandodatei als Kommentar eingetragen.
- Geben Sie den OpenVMS Device Namen für den PDnet-Controller ein (Device Unit). Der Name wird standardmäßig aus der Slot-Nummer abgeleitet (siehe oben).
- Bestätigen sie alle eingegebenen Parameter, um die Kommandoprozedur zum Laden des Gerätetreibers zu erzeugen.
- Beenden Sie die Kommandoprozedur.

Die erzeugte Kommandoprozedur lädt den Gerätetreiber.

3.9.5 Gerätetreiber laden

Bevor der Gerätetreiber geladen werden kann, ist eine manuelle Konfiguration des Kommandos zum Laden des Gerätetreibers notwendig. Da es ist nicht möglich ist, alle notwendigen Informationen zum Laden des Gerätetreibers automatisch zu bestimmen, erfolgt während der Installation des PDnet/AXP kein automatisches Laden. Aus diesem Grunde kann auch die IVP (Installation Verification Procedure) nicht während der Installation ausgeführt werden. Die IVP benötigt einen geladenen Gerätetreiber.

Mit dem von der Kommandoprozedur PDNET$CONFIGURE erzeugtem Kommando, kann der Gerätetreiber geladen werden.

Zum laden des Gerätetreibers gehen Sie wie folgt vor:

- Öffnen Sie ein DECterm Fenster.
- Geben Sie den Befehl ein:

```
$ @SYS$STARTUP:PDNET$STARTUP
```

Nach dem Laden des Gerätetreibers erfolgt die Anzeige des OpenVMS Gerätes (Device Unit).

- Überprüfen Sie den geladenen Gerätetreiber mit dem Befehl:

```
$ show device
```

Wurde der Gerätetreiber erfolgreich geladen, dann muß das OpenVMS Device in der angezeigten Liste existieren, der Status des Gerätes ONLINE und die Anzahl Fehler für dieses Gerät 0 sein. Eine entsprechende Ausgabe sieht so aus:

```
Device                   Device           Error
 Name                    Status           Count
JDA0:                    Online               0
```

Soll das Laden des Gerätetreibers bei jedem System-Startup automatisch erfolgen, dann ist das Kommando zum Laden des Gerätetreibers in die System-Startup-Kommandos einzutragen:

- Öffnen Sie mit einem Texteditor die Datei

  `SYS$SYSROOT:[SYSMGR]SYSTARTUP_VMS.COM`

- Fügen Sie am Dateiende vor dem Befehl EXIT den Befehl ein:

```
$ @SYS$STARTUP:PDNET$STARTUP.COM
```

- Speichern Sie die Datei.
- Beenden Sie den Texteditor.

Beim nächsten Start von OpenVMS lädt das System-Startup-Kommando SYSTARTUP_VMS.COM den Gerätetreiber. Weitere Informationen über die Modifikation des System-Startup Kommandos finden sich im *OpenVMS System Managers Manual*.

ACHTUNG

Achten Sie darauf, das Sie die richtige SYSTARTUP_VMS.COM editieren, falls sich die Datei in mehreren Verzeichnissen befindet.

3.9.6 Aufruf der IVP

Nach dem Laden des Gerätetreibers kann die IVP gestartet werden. Um die IVP aufzurufen, können folgende DCL-Befehle verwendet werden:

```
$ RUN SYS$TEST:PDNET$IVP
```

oder

```
$ PDNETIVP :== $ SYS$TEST:PDNET$IVP
$ PDNETIVP [SHOW]
```

wobei der Parameter SHOW optional ist.

Wurde die IVP erfolgreich ausgeführt, dann wird der Text

```
%PDNET$IVP-I-SUCCESS, IVP successfull done.
```

ausgegeben. Im anderen Fall erfolgt die Ausgabe entsprechender Fehlermeldungen. Bei Angabe des Parameters SHOW erfolgt die Ausgabe von entsprechenden Meldungen, die den Ablauf der IVP anzeigen.

3.10 SUN IPC/IPX (SUN-OS 4.1.2)

Die Kommunikation zwischen dem PAD-SBus und der SUN wird über ein SharedMemoryInterface abgewickelt. Der PAD ist mit 896 KB RAM und 128 KB FlashMemory (gesamt = 1 MByte) ausgestattet, auf welche die SUN über das SharedMemoryInterface zugreifen kann. Der gesamte Speicher des PAD wird linear im Adreßraum der SUN eingeblendet. Daher ist ein Umschalten der PAD-Speichersegmente nicht notwendig.

ACHTUNG

Die PAD-Interface Bibliothek für SUN IPC/IPX (SUN-OS 4.1.2) greift über das SBus-Device auf den PDnet-Controller zu. Dazu muß auf Ihrem System die zugehörige Datei /dev/sbus* vorhanden sein und die Zugriffsrechte für diese Datei gesetzt sein.

3.11 SUN Ultra 5 (Solaris 2.6)

Da beim Betriebssystem Solaris nicht direkt auf die Hardware zugegriffen werden kann, erfolgt der Zugriff auf den PDnet-Controller (PAD-PCI) über einen Gerätetreiber. Dazu muß nach dem Einbau des PDnet-Controllers der Gerätetreiber pdnet auf die Festplatte kopiert und installiert werden. Die verwendeten Ressourcen (Speicherbereich) werden beim PAD-PCI automatisch vergeben. Beim Laden des Gerätetreibers wird jedem installierten PAD-PCI ein Gerätename „`/dev/pdnetX`“ vergeben, wobei für das X die automatisch vergebene Gerätenummer steht.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Dateiname	Beschreibung	Weitergabe
pdnet	Solaris 2.6 Gerätetreiber pdnet für PDnet-Controller PAD-PCI	JA
copydrv	Stapelverarbeitungsdatei zur Installation des Gerätetreibers	JA

Hinweise zur Installation des Gerätetreibers finden Sie unten.

3.11.1 Gerätetreiber installieren

Um den Gerätetreiber pdnet unter Solaris 2.6 zu installieren, müssen sich die Dateien pdnet (Gerätetreiber) und copydrv (Installationsdatei) im gleichen Verzeichnis befinden. Der Gerätetreiber für den PDnet-Controller wird folgendermaßen installiert:

- Öffnen Sie eine Commando-Shell (cmdtool).
- Melden Sie sich als Superuser an.
- Wechsel Sie in das Verzeichnis mit dem Gerätetreiber pdnet.
- Rufen Sie die Installationsdatei auf :
  `./copydrv`

Die Installationsdatei kopiert den Gerätetreiber pdnet in das Verzeichnis: `/usr/kernel/drv/pdnet` und installiert ihn mit dem Befehl:

```
add_drv -m "* 0666 root root" -i '"pci10b5,1168"'
pdnet
```

3.11.2 Gerätetreiber deinstallieren

Um den Gerätetreiber pdnet unter Solaris 2.6 zu deinstallieren:

- Öffnen Sie eine Commando-Shell (cmdtool).
- Melden Sie sich als Superuser an.
- Deinstallieren Sie den Gerätetreiber mit dem Befehl:
  `rem_drv pdnet`
- Löschen Sie den Gerätetreiber mit:
  `rm /usr/kernel/drv/pdnet`

3.12 Motorola-8420 (System V/m88k)

Der PAD-VME ist mit 896 KB RAM und 128 KB FlashMemory (gesamt = 1 MByte) ausgestattet, welches auf dem VME-Bus über ein SharedMemoryInterface eingeblendet wird. Der Speicher des PAD wird linear im VME-Bus Adreßraum eingeblendet.

ACHTUNG

Damit die PAD-Interface Bibliothek für Motorola-8420 (System V/m88k) das SharedMemoryInterface mit der Funktion shmget anlegen kann, muß das mit der PAD-Interface Bibliothek erzeugte Programm im Superuser-Mode ausgeführt werden.

Soll das Programm nicht im Superuser-Mode ausgeführt werden, muß vor dem Programmstart ein anderes Superuser-Programm das SharedMemory angelegt haben und die daraus erhaltende SharedMemory Id shmid als ASCII-Wert in der Datei "pdnetshm.id" im gleichen Verzeichnis wie Ihr Anwenderprogramm ablegen. Wenn diese Datei existiert, liest die PAD-Interface Bibliothek daraus die SharedMemory Id shmid und fordert damit den Zugriff auf das SharedMemory mit der Funktion shmat(shmid, ...) an.

Der Befehl zum Anlegen des SharedMemory lautet:

```
/* VME-PAD ab Adresse 32 MByte (BlockNr 128) */
int vme_page_address = 128 * 256*1024;
int shmid = -1;
shmid = shmget(0, 0x00040000l, 0666 | IPC_CI |
              IPC_CREAT | IPC_PHYS | IPC_NOCLEAR,
              vme_page_address);
if (shmid == -1)  /* Fehler? */
  printf("Fehler bei shmget()!\n");
```

4 Installation Bibliothek

Das folgende Kapitel beschreibt die Dateien der PAD-Interface Bibliothek. Die Bibliothek unterliegt den Copyright Bestimmungen der APEX Automationstechnik GmbH. Nur Dateien, die für die Weitergabe freigegeben sind, dürfen unverändert weiterverkauft werden.

Für die folgenden Plattformen ist die PAD-Interface Bibliothek verfügbar.

4.1 IBM-PC DOS (Real Mode)

Die PAD-Interface Bibliothek für DOS Real Mode Programme ist für folgende Compiler verfügbar.

4.1.1 Borland Pascal 7.0

Die Bibliothek wird in Form von Units (*.TPU) geliefert. Diese Units werden von Borland Pascal in das Programm eingebunden. Damit Borland Pascal die Units einbinden kann, müssen Sie die entsprechenden Dateien von der Diskette in das Unit-Verzeichnis Ihres Projektes kopieren.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
README.TXT	Änderungen und Ergänzungen der gedruckten Dokumentation	NEIN
SRC\PADDEFI.DOK	Interfacedokumentation der Unit PADEFI.TPU	NEIN
SRC\PADITF.DOK	Interfacedokumentation der Unit PADITF.TPU	NEIN
SRC\PADLIB.DOK	Interfacedokumentation der Unit PADLIB.TPU	NEIN
SRC\PADLIB.DOK	Interfacedokumentation der Unit PADVDM.TPU	NEIN
SRC\PADTEST.PAS	Quellcode für Beispielprogramm	NEIN

Verzeichnis\Dateiname	Beschreibung	Weitergabe
DOSBP70\PADDEFI.TPU	Basis Typdefinitionen	NEIN
DOSBP70\PADAPIPC.TPU	Interface zum DOS-PC	NEIN
DOSBP70\PADITF.TPU	Funktionen PAD_???	NEIN
DOSBP70\PADLIB.TPU	Funktionen PDnet_???	NEIN
DOSBP70\PADVDM.TPU	Funktionen VDM_???	NEIN
DOSBP70\BP.TP	Compiler Konfigurationsdatei für Beispielprogramm	NEIN
DOSBP70\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

In der Beschreibung wird nicht bei jeder Funktion, Variablen o.ä. angegeben, in welcher Unit diese definiert ist, da eine logische Zuordnung über Ihren Namen bzw. der Funktionsgruppe ersichtlich ist.

4.1.2 Borland C++ 3.1

Die Bibliothek wird in Form von Objektdateien (*.OBJ) geliefert. Diese Dateien werden von Borland C++ 3.1 in das Programm eingebunden. Damit der C-Compiler die Dateien einbinden kann, müssen Sie die entsprechenden Dateien von der Diskette in das Objektdateiverzeichnis Ihres Projektes kopieren.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
README.TXT	Änderungen und Ergänzungen der gedruckten Dokumentation	NEIN
SRC\PADDEFI.H	Headerdatei für Basis Typdefinitionen	NEIN
SRC\PADITF.H	Headerdatei für Funktionen PAD_???	NEIN
SRC\PADLIB.H	Headerdatei für Funktionen PDnet_???	NEIN
SRC\PADVDM.H	Headerdatei für Funktionen VDM_???	NEIN
SRC\PADTEST.H	Headerdatei für Beispielprogramm	NEIN
SRC\PADTEST.C	Quellcode für Beispielprogramm - 1. Teil	NEIN
SRC\PADTEST1.C	Quellcode für Beispielprogramm - 2. Teil	NEIN
SRC\PADTEST2.C	Quellcode für Beispielprogramm - 3. Teil	NEIN
DOSBC31\PADAPIPC.OBJ	Interface zum DOS-PC	NEIN
DOSBC31\PADITF.OBJ	Funktionen PAD_???	NEIN
DOSBC31\PADLIB.OBJ	Funktionen PDnet_???	NEIN
DOSBC31\PADVDM.OBJ	Funktionen VDM_???	NEIN
DOSBC31\PADTEST.PRJ	Projektdatei für Beispielprogramm	NEIN
DOSBC31\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

In der Beschreibung wird nicht bei jeder Funktion, Variablen o.ä. angegeben, in welcher Bibliothek diese definiert ist, da eine logische Zuordnung über Ihren Namen bzw. der Funktionsgruppe ersichtlich ist.

4.2 IBM-PC DOS (Protected Mode)

Die PAD-Interface Bibliothek für DOS Protected Mode Programme ist für folgende Compiler verfügbar.

4.2.1 Borland Pascal 7.0

Die Bibliothek wird in Form einer Unit (*.PAS) und einer dynamischen Linkbibliothek (*.DLL) geliefert. Damit Borland Pascal 7.0 auf die DLL zugreifen kann, kopieren Sie die Unit PADITF16.PAS in Ihr Projektverzeichnis und die Datei PADITF16.DLL in das Verzeichnis Ihres Anwenderprogramms. Binden Sie danach die Unit PADITF16.PAS in Ihr Programm ein.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
README.TXT	Änderungen und Ergänzungen der gedruckten Dokumentation	NEIN
SRC\PADITF16.PAS	Interface der PADITF16.DLL	NEIN
SRC\PADTEST.PAS	Quellcode für Beispielprogramm	NEIN
DPMIBP70\PADITF16.DLL	Dynamische Linkbibliothek mit Funktionen	JA
DPMIBP70\PADITF16.TPP	Funktionen PAD_???, PDnet_???, VDM_???	NEIN
DPMIBP70\BP.TP	Compiler Konfigurationsdatei für Beispielprogramm	NEIN
DPMIBP70\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

Die PADITF16.DLL ist nicht reentrantfähig, da sie globale Variablen benutzt. Deshalb darf nur ein Programm gleichzeitig auf die PADITF16.DLL zugreifen.

4.3 IBM-PC Windows 3.1

Die PAD-Interface Bibliothek für Windows 3.1 Programme ist für folgende Compiler verfügbar.

4.3.1 Borland Pascal 7.0

Die Bibliothek wird in Form einer Unit (*.PAS) und einer dynamischen Linkbibliothek (*.DLL) geliefert. Damit Borland Pascal 7.0 auf die DLL zugreifen kann, kopieren Sie die Unit PADITF16.PAS in Ihr Projektverzeichnis und die Datei PADITF16.DLL in das Verzeichnis Ihres Anwenderprogramms. Binden Sie danach die Unit PADITF16.PAS in Ihr Programm ein.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
README.TXT	Änderungen und Ergänzungen der gedruckten Dokumentation	NEIN
SRC\PADITF16.PAS	Interface der PADITF16.DLL	NEIN
SRC\PADTEST.PAS	Quellcode für Beispielprogramm	NEIN
W31BP70\PADITF16.DLL	Dynamische Linkbibliothek mit Funktionen PAD_???, PDnet_???, VDM_???	JA
W31BP70\PADITF16.TPW	Funktionen PAD_???, PDnet_???, VDM_???	NEIN
W31BP70\BP.TP	Compiler Konfigurationsdatei für Beispielprogramm	NEIN
W31BP70\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

Die PADITF16.DLL ist nicht reentrantfähig, da sie globale Variablen benutzt. Deshalb darf nur ein Programm gleichzeitig auf die PADITF16.DLL zugreifen.

4.3.2 Borland C++ 3.1

Die Bibliothek wird in Form einer Importbibliothek (*.LIB) und einer dynamischen Linkbibliothek (*.DLL) geliefert. Damit Borland C++ 3.1 auf die DLL zugreifen kann, kopieren Sie die Importbibliothek PADITF16.LIB in Ihr Projektverzeichnis und die Datei PADITF16.DLL in das Verzeichnis Ihres Anwenderprogramms. Binden Sie danach die Importbibliothek PADITF16.LIB und die Headerdatei PAD2DLL.H in Ihr Programm ein.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
README.TXT	Änderungen und Ergänzungen der gedruckten Dokumentation	NEIN
SRC\PAD2DLL.H	Headerdatei für Anpassung an PADITF16.H	NEIN
SRC\PADITF16.H	Headerdatei für PADITF16.DLL	NEIN
SRC\PADDEFI.H	Headerdatei für Basis Typdefinitionen	NEIN
SRC\PADITF.H	Headerdatei für Funktionen PAD_???	NEIN
SRC\PADLIB.H	Headerdatei für Funktionen PDnet_???	NEIN
SRC\PADVDM.H	Headerdatei für Funktionen VDM_???	NEIN
SRC\PADTEST.H	Headerdatei für Beispielprogramm	NEIN
SRC\PADTEST.C	Quellcode für Beispielprogramm - 1. Teil	NEIN
SRC\PADTEST1.C	Quellcode für Beispielprogramm - 2. Teil	NEIN
SRC\PADTEST2.C	Quellcode für Beispielprogramm - 3. Teil	NEIN
W31BC31\PADITF16DLL	Dynamische Linkbibliothek mit Funktionen	JA
W31BC31\PADITF16.LIB	Importbibliothek für PADITF16.DLL	NEIN
W31BC31\PADTEST.PRJ	Projektdatei für Beispielprogramm	NEIN
W31BC31\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

Die PADITF16.DLL ist nicht reentrantfähig, da sie globale Variablen benutzt. Deshalb darf nur ein Programm gleichzeitig auf die PADITF16.DLL zugreifen.

4.4 IBM-PC Windows 95

Die PAD-Interface Bibliothek für Windows 95 Programme ist für folgende Compiler verfügbar.

4.4.1 Borland Delphi 2.0

Die Bibliothek wird in Form einer Unit (*.PAS) und einer dynamischen Linkbibliothek (*.DLL) geliefert. Damit Borland Delphi 2.0 auf die DLL zugreifen kann, kopieren Sie die Unit PADITF32.PAS in Ihr Projektverzeichnis und die Datei PADITF32.DLL in das Verzeichnis Ihres Anwenderprogramms. Binden Sie danach die Unit PADITF32.PAS in Ihr Programm ein.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
README.TXT	Änderungen und Ergänzungen der gedruckten Dokumentation	NEIN
SRC\PADITF32.PAS	Interface der PADITF32.DLL	NEIN
SRC\PADTEST.DPR	Quellcode für Beispielprogramm	NEIN
DELPHI20\PADITF32.DLL	Dynamische Linkbibliothek mit Funktionen PAD_???, PDnet_???, VDM_???	JA
DELPHI20\PADITF32.DCU	Funktionen PAD_???, PDnet_???, VDM_???	NEIN
DELPHI20\PADITF32.RES	Ressourcendatei für Beispielprogramm	NEIN
DELPHI20\PADTEST.DOF	Compiler Konfigurationsdatei für Beispielprogramm	NEIN
DELPHI20\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

Der Zugriff der DLL auf den PAD erfolgt über den Gerätetreiber PADITF32.VXD. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

4.4.2 Borland C++ 4.5

Die Bibliothek wird in Form einer Importbibliothek (*.LIB) und einer dynamischen Linkbibliothek (*.DLL) geliefert. Damit Borland C++ 4.5 auf die DLL zugreifen kann, kopieren Sie die Importbibliothek PADITF32.LIB in Ihr Projektverzeichnis und die Datei PADITF32.DLL in das Verzeichnis Ihres Anwenderprogramms. Binden Sie danach die Importbibliothek PADITF32.LIB und die Headerdatei PAD2DLL.H in Ihr Programm ein.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
SRC\PAD2DLL.H	Headerdatei für Anpassung an PADITF32.H	NEIN
SRC\PADITF32.H	Headerdatei für PADITF32.DLL	NEIN
SRC\PADDEFI.H	Headerdatei für Basis Typdefinitionen	NEIN
SRC\PADITF.H	Headerdatei für Funktionen PAD_???	NEIN
SRC\PADLIB.H	Headerdatei für Funktionen PDnet_???	NEIN
SRC\PADVDM.H	Headerdatei für Funktionen VDM_???	NEIN
SRC\PADTEST.H	Headerdatei für Beispielprogramm	NEIN
SRC\PADTEST.C	Quellcode für Beispielprogramm - 1. Teil	NEIN
SRC\PADTEST1.C	Quellcode für Beispielprogramm - 2. Teil	NEIN
SRC\PADTEST2.C	Quellcode für Beispielprogramm - 3. Teil	NEIN
W95BC45\PADITF32.DLL	Dynamische Linkbibliothek mit Funktionen PAD_???, PDnet_???, VDM_???	JA
W95BC45\PADITF32.LIB	Importbibliothek für PADITF32.DLL	NEIN
W95BC45\PADTEST.DIE	Projektdatei für Beispielprogramm	NEIN
W95BC45\PADTEST.DSW	Kontextdatei für Beispielprogramm	NEIN
W95BC45\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

Der Zugriff der DLL auf den PAD erfolgt über den Gerätetreiber PADITF32.VXD. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

4.4.3 Microsoft Visual C++ 2.0

Die Bibliothek wird in Form einer Importbibliothek (*.LIB) und einer dynamischen Linkbibliothek (*.DLL) geliefert. Damit Visual C++ 2.0 auf die DLL zugreifen kann, kopieren Sie die Importbibliothek PADITF32.LIB in Ihr Projektverzeichnis und die Datei PADITF32.DLL in das Verzeichnis Ihres Anwenderprogramms. Binden Sie danach die Importbibliothek PADITF32.LIB und die Headerdatei PAD2DLL.H in Ihr Programm ein.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
SRC\PAD2DLL.H	Headerdatei für Anpassung an PADITF32.H	NEIN
SRC\PADITF32.H	Headerdatei für PADITF32.DLL	NEIN
SRC\PADDEFI.H	Headerdatei für Basis Typdefinitionen	NEIN
SRC\PADITF.H	Headerdatei für Funktionen PAD_???	NEIN
SRC\PADLIB.H	Headerdatei für Funktionen PDnet_???	NEIN
SRC\PADVDM.H	Headerdatei für Funktionen VDM_???	NEIN
SRC\PADTEST.H	Headerdatei für Beispielprogramm	NEIN
SRC\PADTEST.C	Quellcode für Beispielprogramm - 1. Teil	NEIN
SRC\PADTEST1.C	Quellcode für Beispielprogramm - 2. Teil	NEIN
SRC\PADTEST2.C	Quellcode für Beispielprogramm - 3. Teil	NEIN
W95MSVC2\PADITF32.DLL	Dynamische Linkbibliothek mit Funktionen PAD_???, PDnet_???, VDM_???	JA
W95MSVC2\PADITF32.LIB	Importbibliothek für PADITF32.DLL	NEIN
W95MSVC2\PADTEST.VCP	Projektdatei für Beispielprogramm	NEIN
W95MSVC2\PADTEST.MAK	Makedatei für Beispielprogramm	NEIN
W95MSVC2\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

Der Zugriff der DLL auf den PAD erfolgt über den Gerätetreiber PADITF32.VXD. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändert-flags löschen.

4.4.4 Microsoft Visual C++ 4.0

Die Bibliothek wird in Form einer Importbibliothek (*.LIB) und einer dynamischen Linkbibliothek (*.DLL) geliefert. Damit Visual C++ 4.0 auf die DLL zugreifen kann, kopieren Sie die Importbibliothek PADITF32.LIB in Ihr Projektverzeichnis und die Datei PADITF32.DLL in das Verzeichnis Ihres Anwenderprogramms. Binden Sie danach die Importbibliothek PADITF32.LIB und die Headerdatei PAD2DLL.Hin Ihr Programm ein.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
SRC\PAD2DLL.H	Headerdatei für Anpassung an PADITF32.H	NEIN
SRC\PADITF32.H	Headerdatei für PADITF32.DLL	NEIN
SRC\PADDEFI.H	Headerdatei für Basis Typdefinitionen	NEIN
SRC\PADITF.H	Headerdatei für Funktionen PAD_???	NEIN
SRC\PADLIB.H	Headerdatei für Funktionen PDnet_???	NEIN
SRC\PADVDM.H	Headerdatei für Funktionen VDM_???	NEIN
SRC\PADTEST.H	Headerdatei für Beispielprogramm	NEIN
SRC\PADTEST.C	Quellcode für Beispielprogramm - 1. Teil	NEIN
SRC\PADTEST1.C	Quellcode für Beispielprogramm - 2. Teil	NEIN
SRC\PADTEST2.C	Quellcode für Beispielprogramm - 3. Teil	NEIN
WNTMSVC4\PADITF32.DLL	Dynamische Linkbibliothek mit Funktionen PAD_???, PDnet_???, VDM_???	JA
W95MSVC4\PADITF32.LIB	Importbibliothek für PADITF32.DLL	NEIN
W95MSVC4\PADTEST.MDP	Projektdatei für Beispielprogramm	NEIN
W95MSVC4\PADTEST.MAK	Makedatei für Beispielprogramm	NEIN
W95MSVC4\PADTEST.NCB	Programm Database für Beispielprogramm	NEIN
W95MSVC4\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

Der Zugriff der DLL auf den PAD erfolgt über den Gerätetreiber PADITF32.VXD. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

4.4.5 Watcom C++ 10.0

Die Bibliothek wird in Form einer Importbibliothek (*.LIB) und einer dynamischen Linkbibliothek (*.DLL) geliefert. Damit Watcom C++ 10.0 auf die DLL zugreifen kann, kopieren Sie die Importbibliothek PADITF32.LIB in Ihr Projektverzeichnis und die Datei PA-DITF32.DLL in das Verzeichnis Ihres Anwenderprogramms. Binden Sie danach die Importbibliothek PADITF32.LIB in Ihr Programm ein.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
SRC\PAD2DLL.H	Headerdatei für Anpassung an PADITF32.H	NEIN
SRC\PADITF32.H	Headerdatei für PADITF32.DLL	NEIN
SRC\PADDEFI.H	Headerdatei für Basis Typdefinitionen	NEIN
SRC\PADITF.H	Headerdatei für Funktionen PAD_???	NEIN
SRC\PADLIB.H	Headerdatei für Funktionen PDnet_???	NEIN
SRC\PADVDM.H	Headerdatei für Funktionen VDM_???	NEIN
SRC\PADTEST.H	Headerdatei für Beispielprogramm	NEIN
SRC\PADTEST.C	Quellcode für Beispielprogramm - 1. Teil	NEIN
SRC\PADTEST1.C	Quellcode für Beispielprogramm - 2. Teil	NEIN
SRC\PADTEST2.C	Quellcode für Beispielprogramm - 3. Teil	NEIN
W95WAT10\PADITF32.DLL	Dynamische Linkbibliothek mit Funktionen PAD_???, PDnet_???, VDM_???	JA
W95WAT10\PADITF32.LIB	Importbibliothek für PA-DITF32.DLL	NEIN
W95WAT10\PADTEST.WPJ	Projektdatei für Beispielprogramm	NEIN
W95WAT10\PADTEST.LK1	Linkdatei für Beispielprogramm	NEIN
W95WAT10\PADTEST.MK	Makedatei für Beispielprogramm	NEIN
W95WAT10\PADTEST.MK1	Makedatei für Beispielprogramm	NEIN
W95WAT10\PADTEST.TGT	Targetdatei für Beispielprogramm	NEIN
W95WAT10\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

Der Zugriff der DLL auf den PAD erfolgt über den Gerätetreiber PADITF32.VXD. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändert-flags löschen.

4.5 IBM-PC Windows NT 3.5x/4.00

Die PAD-Interface Bibliothek für Windows NT 3.5x/4.00 Programme ist für folgende Compiler verfügbar.

4.5.1 Borland Delphi 2.0

Die Bibliothek wird in Form einer Unit (*.PAS) und einer dynamischen Linkbibliothek (*.DLL) geliefert. Damit Borland Delphi 2.0 auf die DLL zugreifen kann, kopieren Sie die Unit PADITF32.PAS in Ihr Projektverzeichnis und die Datei PADITF32.DLL in das Verzeichnis Ihres Anwenderprogramms. Binden Sie danach die Unit PADITF32.PAS in Ihr Programm ein.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
README.TXT	Änderungen und Ergänzungen der gedruckten Dokumentation	NEIN
SRC\PADITF32.PAS	Interface der PADITF32.DLL	NEIN
SRC\PADTEST.DPR	Quellcode für Beispielprogramm	NEIN
DELPHI20\PADITF32.DLL	Dynamische Linkbibliothek mit Funktionen PAD_???, PDnet_???, VDM_???	JA
DELPHI20\PADITF32.DCU	Funktionen PAD_???, PDnet_???, VDM_???	NEIN
DELPHI20\PADITF32.RES	Ressourcendatei für Beispielprogramm	NEIN
DELPHI20\PADTEST.DOF	Compiler Konfigurationsdatei für Beispielprogramm	NEIN
DELPHI20\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

Der Zugriff der DLL auf den PAD erfolgt über den Gerätetreiber PADITF32.SYS. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

4.5.2 Borland C++ 4.5

Die Bibliothek wird in Form einer Importbibliothek (*.LIB) und einer dynamischen Linkbibliothek (*.DLL) geliefert. Damit Borland C++ 4.5 auf die DLL zugreifen kann, kopieren Sie die Importbibliothek PADITF32.LIB in Ihr Projektverzeichnis und die Datei PADITF32.DLL in das Verzeichnis Ihres Anwenderprogramms. Binden Sie danach die Importbibliothek PADITF32.LIB und die Headerdatei PAD2DLL.H in Ihr Programm ein.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
SRC\PAD2DLL.H	Headerdatei für Anpassung an PADITF32.H	NEIN
SRC\PADITF32.H	Headerdatei für PADITF32.DLL	NEIN
SRC\PADDEFI.H	Headerdatei für Basis Typdefinitionen	NEIN
SRC\PADITF.H	Headerdatei für Funktionen PAD_???	NEIN
SRC\PADLIB.H	Headerdatei für Funktionen PDnet_???	NEIN
SRC\PADVDM.H	Headerdatei für Funktionen VDM_???	NEIN
SRC\PADTEST.H	Headerdatei für Beispielprogramm	NEIN
SRC\PADTEST.C	Quellcode für Beispielprogramm - 1. Teil	NEIN
SRC\PADTEST1.C	Quellcode für Beispielprogramm - 2. Teil	NEIN
SRC\PADTEST2.C	Quellcode für Beispielprogramm - 3. Teil	NEIN
WNTBC45\PADITF32.DLL	Dynamische Linkbibliothek mit Funktionen PAD_???, PDnet_???, VDM_???	JA
WNTBC45\PADITF32.LIB	Importbibliothek für PADITF32.DLL	NEIN
WNTBC45\PADTEST.DIE	Projektdatei für Beispielprogramm	NEIN
WNTBC45\PADTEST.DSW	Contextdatei für Beispielprogramm	NEIN
WNTBC45\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

Der Zugriff der DLL auf den PAD erfolgt über den Gerätetreiber PADITF32.SYS. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

4.5.3 Microsoft Visual C++ 2.0

Die Bibliothek wird in Form einer Importbibliothek (*.LIB) und einer dynamischen Linkbibliothek (*.DLL) geliefert. Damit Visual C++ 2.0 auf die DLL zugreifen kann, kopieren Sie die Importbibliothek PADITF32.LIB in Ihr Projektverzeichnis und die Datei PADITF32.DLL in das Verzeichnis Ihres Anwenderprogramms. Binden Sie danach die Importbibliothek PADITF32.LIB und die Headerdatei PAD2DLL.H in Ihr Programm ein.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
SRC\PAD2DLL.H	Headerdatei für Anpassung an PADITF32.H	NEIN
SRC\PADITF32.H	Headerdatei für PADITF32.DLL	NEIN
SRC\PADDEFI.H	Headerdatei für Basis Typdefinitionen	NEIN
SRC\PADITF.H	Headerdatei für Funktionen PAD_???	NEIN
SRC\PADLIB.H	Headerdatei für Funktionen PDnet_???	NEIN
SRC\PADVDM.H	Headerdatei für Funktionen VDM_???	NEIN
SRC\PADTEST.H	Headerdatei für Beispielprogramm	NEIN
SRC\PADTEST.C	Quellcode für Beispielprogramm - 1. Teil	NEIN
SRC\PADTEST1.C	Quellcode für Beispielprogramm - 2. Teil	NEIN
SRC\PADTEST2.C	Quellcode für Beispielprogramm - 3. Teil	NEIN
WNTMSVC2\PADITF32.DLL	Dynamische Linkbibliothek mit Funktionen PAD_???, PDnet_???, VDM_???	JA
WNTMSVC2\PADITF32.LIB	Importbibliothek für PADITF32.DLL	NEIN
WNTMSVC2\PADTEST.VCP	Projektdatei für Beispielprogramm	NEIN
WNTMSVC2\PADTEST.MAK	Makedatei für Beispielprogramm	NEIN
WNTMSVC2\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

Der Zugriff der DLL auf den PAD erfolgt über den Gerätetreiber PADITF32.SYS. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

4.5.4 Microsoft Visual C++ 4.0

Die Bibliothek wird in Form einer Importbibliothek (*.LIB) und einer dynamischen Linkbibliothek (*.DLL) geliefert. Damit Visual C++ 4.0 auf die DLL zugreifen kann, kopieren Sie die Importbibliothek PA-DITF32.LIB in Ihr Projektverzeichnis und die Datei PADITF32.DLL in das Verzeichnis Ihres Anwenderprogramms. Binden Sie danach die Importbibliothek PADITF32.LIB und die Headerdatei PAD2DLL.H in Ihr Programm ein.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
SRC\PAD2DLL.H	Headerdatei für Anpassung an PADITF32.H	NEIN
SRC\PADITF32.H	Headerdatei für PADITF32.DLL	NEIN
SRC\PADDEFI.H	Headerdatei für Basis Typdefinitionen	NEIN
SRC\PADITF.H	Headerdatei für Funktionen PAD_???	NEIN
SRC\PADLIB.H	Headerdatei für Funktionen PDnet_???	NEIN
SRC\PADVDM.H	Headerdatei für Funktionen VDM_???	NEIN
SRC\PADTEST.H	Headerdatei für Beispielprogramm	NEIN
SRC\PADTEST.C	Quellcode für Beispielprogramm - 1. Teil	NEIN
SRC\PADTEST1.C	Quellcode für Beispielprogramm - 2. Teil	NEIN
SRC\PADTEST2.C	Quellcode für Beispielprogramm - 3. Teil	NEIN
WNTMSVC4\PADITF32.DLL	Dynamische Linkbibliothek mit Funktionen PAD_???, PDnet_???, VDM_???	JA
WNTMSVC4\PADITF32.LIB	Importbibliothek für PADITF32.DLL	NEIN
WNTMSVC4\PADTEST.MDP	Projektdatei für Beispielprogramm	NEIN
WNTMSVC4\PADTEST.MAK	Makedatei für Beispielprogramm	NEIN
WNTMSVC4\PADTEST.NCB	Programm Database für Beispielprogramm	NEIN
WNTMSVC4\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

Der Zugriff der DLL auf den PAD erfolgt über den Gerätetreiber PADITF32.SYS. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändert-flags löschen.

4.5.5 Watcom C++ 10.0

Die Bibliothek wird in Form einer Importbibliothek (*.LIB) und einer dynamischen Linkbibliothek (*.DLL) geliefert. Damit Watcom C++ 10.0 auf die DLL zugreifen kann, kopieren Sie die Importbibliothek PADITF32.LIB in Ihr Projektverzeichnis und die Datei PADITF32.DLL in das Verzeichnis Ihres Anwenderprogramms. Binden Sie danach die Importbibliothek PADITF32.LIB in Ihr Programm ein.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
SRC\PAD2DLL.H	Headerdatei für Anpassung an PADITF32.H	NEIN
SRC\PADITF32.H	Headerdatei für PADITF32.DLL	NEIN
SRC\PADDEFI.H	Headerdatei für Basis Typdefinitionen	NEIN
SRC\PADITF.H	Headerdatei für Funktionen PAD_???	NEIN
SRC\PADLIB.H	Headerdatei für Funktionen PDnet_???	NEIN
SRC\PADVDM.H	Headerdatei für Funktionen VDM_???	NEIN
SRC\PADTEST.H	Headerdatei für Beispielprogramm	NEIN
SRC\PADTEST.C	Quellcode für Beispielprogramm - 1. Teil	NEIN
SRC\PADTEST1.C	Quellcode für Beispielprogramm - 2. Teil	NEIN
SRC\PADTEST2.C	Quellcode für Beispielprogramm - 3. Teil	NEIN
WNTWAT10\PADITF32.DLL	Dynamische Linkbibliothek mit Funktionen PAD_???, PDnet_???, VDM_???	JA
WNTWAT10\PADITF32.LIB	Importbibliothek für PADITF32.DLL	NEIN
WNTWAT10\PADTEST.WPJ	Projektdatei für Beispielprogramm	NEIN
WNTWAT10\PADTEST.LK1	Linkdatei für Beispielprogramm	NEIN
WNTWAT10\PADTEST.MK	Makedatei für Beispielprogramm	NEIN
WNTWAT10\PADTEST.MK1	Makedatei für Beispielprogramm	NEIN
WNTWAT10\PADTEST.TGT	Targetdatei für Beispielprogramm	NEIN
WNTWAT10\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

Der Zugriff der DLL auf den PAD erfolgt über den Gerätetreiber PADITF32.SYS. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

4.6 IBM-PC OS/2 Warp 3/4

Die PAD-Interface Bibliothek für OS/2 Warp Programme ist für folgende Compiler verfügbar.

4.6.1 Borland C++ 2.0 für OS/2

Die Bibliothek wird in Form einer Importbibliothek (*.LIB) und einer dynamischen Linkbibliothek (*.DLL) geliefert. Damit Borland C++ 2.0 auf die DLL zugreifen kann, kopieren Sie die Importbibliothek PADITF32.LIB in Ihr Projektverzeichnis und die Datei PADITF32.DLL in das Verzeichnis Ihres Anwenderprogramms. Binden Sie danach die Importbibliothek PADITF32.LIB und die Headerdatei PAD2DLL.H in Ihr Programm ein.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
README.TXT	Änderungen und Ergänzungen der gedruckten Dokumentation	NEIN
SRC\PAD2DLL.H	Headerdatei für Anpassung an PADITF32.H	NEIN
SRC\PADITF32.H	Headerdatei für PADITF32.DLL	NEIN
SRC\PADDEFI.H	Headerdatei für Basis Typdefinitionen	NEIN
SRC\PADITF.H	Headerdatei für Funktionen PAD_???	NEIN
SRC\PADLIB.H	Headerdatei für Funktionen PDnet_???	NEIN
SRC\PADVDM.H	Headerdatei für Funktionen VDM_???	NEIN
SRC\PADTEST.H	Headerdatei für Beispielprogramm	NEIN
SRC\PADTEST.C	Quellcode für Beispielprogramm - 1. Teil	NEIN
SRC\PADTEST1.C	Quellcode für Beispielprogramm - 2. Teil	NEIN
SRC\PADTEST2.C	Quellcode für Beispielprogramm - 3. Teil	NEIN
OS2BC20\PADITF32.DLL	Dynamische Linkbibliothek mit Funktionen PAD_???, PDnet_???, VDM_???	JA
OS2BC20\PADITF32.LIB	Importbibliothek für PADITF32.DLL	NEIN
OS2BC20\PADTEST.PRJ	Projektdatei für Beispielprogramm	NEIN
OS2BC20\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

Der Zugriff der DLL auf den PAD erfolgt über den Einheitentreiber APEX-PAD.SYS. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-

Speicher zugreifen, da sie sich sonst gegenseitig die Geändert-flags löschen.

4.6.2 VisualAge C++ 3.0

Die Bibliothek wird in Form einer Importbibliothek (*.LIB) und einer dynamischen Linkbibliothek (*.DLL) geliefert. Damit VisualAge C++ 3.0 auf die DLL zugreifen kann, kopieren Sie die Importbibliothek PADITF32.LIB in Ihr Projektverzeichnis und die Datei PA-DITF32.DLL in das Verzeichnis Ihres Anwenderprogramms. Bin-den Sie danach die Importbibliothek PADITF32.LIB und die Hea-derdatei PAD2DLL.H in Ihr Programm ein.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
SRC\PAD2DLL.H	Headerdatei für Anpassung an PADITF32.H	NEIN
SRC\PADITF32.H	Headerdatei für PADITF32.DLL	NEIN
SRC\PADDEFI.H	Headerdatei für Basis Typdefini-tionen	NEIN
SRC\PADITF.H	Headerdatei für Funktionen PAD_???	NEIN
SRC\PADLIB.H	Headerdatei für Funktionen PDnet_???	NEIN
SRC\PADVDM.H	Headerdatei für Funktionen VDM_???	NEIN
SRC\PADTEST.H	Headerdatei für Beispielpro-gramm	NEIN
SRC\PADTEST.C	Quellcode für Beispielprogramm - 1. Teil	NEIN
SRC\PADTEST1.C	Quellcode für Beispielprogramm - 2. Teil	NEIN
SRC\PADTEST2.C	Quellcode für Beispielprogramm - 3. Teil	NEIN
OS2VAGE3\PADITF32.DLL	Dynamische Linkbibliothek mit Funktionen PAD_???, PDnet_???, VDM_???	JA
OS2VAGE3\PADITF32.LIB	Importbibliothek für PA-DITF32.DLL	NEIN
OS2VAGE3\PADTEST	Projektdatei für Beispielpro-gramm	NEIN
OS2VAGE3\PADTEST.MAK	Makedatei für Beispielprogramm	NEIN
OS2VAGE3\PADTEST.$MM	Make-Makedatei für Beispielpro-gramm	NEIN
OS2VAGE3\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

Der Zugriff der DLL auf den PAD erfolgt über den Einheitentreiber APEX-PAD.SYS. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändert-flags löschen.

4.6.3 Watcom C++ 10.0

Die Bibliothek wird in Form einer Importbibliothek (*.LIB) und einer dynamischen Linkbibliothek (*.DLL) geliefert. Damit Watcom C++ 10.0 auf die DLL zugreifen kann, kopieren Sie die Importbibliothek PADITF32.LIB in Ihr Projektverzeichnis und die Datei PADITF32.DLL in das Verzeichnis Ihres Anwenderprogramms. Binden Sie danach die Importbibliothek PADITF32.LIB in Ihr Programm ein.

Folgende Dateien befinden sich auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
SRC\PAD2DLL.H	Headerdatei für Anpassung an PADITF32.H	NEIN
SRC\PADITF32.H	Headerdatei für PADITF32.DLL	NEIN
SRC\PADDEFI.H	Headerdatei für Basis Typdefinitionen	NEIN
SRC\PADITF.H	Headerdatei für Funktionen PAD_???	NEIN
SRC\PADLIB.H	Headerdatei für Funktionen PDnet_???	NEIN
SRC\PADVDM.H	Headerdatei für Funktionen VDM_???	NEIN
SRC\PADTEST.H	Headerdatei für Beispielprogramm	NEIN
SRC\PADTEST.C	Quellcode für Beispielprogramm - 1. Teil	NEIN
SRC\PADTEST1.C	Quellcode für Beispielprogramm - 2. Teil	NEIN
SRC\PADTEST2.C	Quellcode für Beispielprogramm - 3. Teil	NEIN
OS2WAT10\PADITF32.DLL	Dynamische Linkbibliothek mit Funktionen PAD_???, PDnet_???, VDM_???	JA
OS2WAT10\PADITF32.LIB	Importbibliothek für PADITF32.DLL	NEIN
OS2WAT10\PADTEST.WPJ	Projektdatei für Beispielprogramm	NEIN
OS2WAT10\PADTEST.LK1	Linkdatei für Beispielprogramm	NEIN
OS2WAT10\PADTEST.MK	Makedatei für Beispielprogramm	NEIN
OS2WAT10\PADTEST.MK1	Makedatei für Beispielprogramm	NEIN
OS2WAT10\PADTEST.TGT	Targetdatei für Beispielprogramm	NEIN
OS2WAT10\PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

Der Zugriff der DLL auf den PAD erfolgt über den Einheitentreiber APEX-PAD.SYS. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

4.7 IBM-PC QNX 3.21 (Protected Mode)

Die PAD-Interface Bibliothek für QNX 3.21 (Protected Mode) Programme ist für folgende Compiler verfügbar.

4.7.1 C86 für QNX

Die Bibliothek wird in Form von Objektdateien (*.OBJ) geliefert. Diese Dateien werden von C86 für QNX (cq) in das Programm eingebunden. Damit der C-Compiler die Dateien einbinden kann, müssen Sie die entsprechenden Dateien von der Diskette in das Verzeichnis Ihres Projektes kopieren.

Folgende Dateien befinden sich im Root-Verzeichnis auf der mitgelieferten Diskette:

Dateiname	Beschreibung	Weitergabe
readme.txt	Änderungen und Ergänzungen der gedruckten Dokumentation	NEIN
paditf.h	Headerdatei für Funktionen PAD_???	NEIN
padlib.h	Headerdatei für Funktionen PDnet_???	NEIN
padtest.h	Headerdatei für Beispielprogramm	NEIN
padtest.c	Quellcode für Beispielprogramm - 1. Teil	NEIN
padtest2.c	Quellcode für Beispielprogramm - 2. Teil	NEIN
paditf.obj	Funktionen PAD_???	NEIN
padlib.obj	Funktionen PDnet_??? und VDM_???	NEIN
makefile	Batch-Datei zum Compilieren des Beispielprogramms	NEIN
padtest	Ausführbares Beispielprogramm	JA

HINWEIS

In der Beschreibung wird nicht bei jeder Funktion, Variablen o.ä. angegeben, in welcher Bibliothek diese definiert ist, da eine logische Zuordnung über Ihren Namen bzw. der Funktionsgruppe ersichtlich ist.

4.8 IBM-PC QNX 4.22 (Protected Mode)

Die PAD-Interface Bibliothek für QNX 4.22 (Protected Mode) Programme ist für folgende Compiler verfügbar.

4.8.1 Watcom C 9.5

Die Bibliothek wird in Form von Objektdateien (*.O) geliefert. Diese Dateien werden von Watcom C (wcc386) in das Programm eingebunden. Damit der C-Compiler die Dateien einbinden kann, müssen Sie die entsprechenden Dateien von der Diskette in das Verzeichnis Ihres Projektes kopieren.

Die Lieferdiskette hat das MS-DOS Dateiformat. Deshalb müssen sie das DOS Dateisystem starten, bevor Sie die Dateien kopieren:

- `# Dosfsys`
- `# cp /dos/a/* .`

Folgende Dateien befinden sich im Root-Verzeichnis auf der mitgelieferten Diskette:

Dateiname	Beschreibung	Weitergabe
readme.txt	Änderungen und Ergänzungen der gedruckten Dokumentation	NEIN
paddefi.h	Headerdatei für Basistypen	NEIN
paditf.h	Headerdatei für Funktionen PAD_???	NEIN
padlib.h	Headerdatei für Funktionen PDnet_???	NEIN
padvdm.h	Headerdatei für Funktionen VDM_???	NEIN
padtest.h	Headerdatei für Beispielprogramm	NEIN
padtest.c	Quellcode für Beispielprogramm - 1. Teil	NEIN
padtest1.c	Quellcode für Beispielprogramm - 2. Teil	NEIN
padtest2.c	Quellcode für Beispielprogramm - 3. Teil	NEIN
padapiq4.obj	Interface für QNX 4.22	NEIN
paditf.obj	Funktionen PAD_???	NEIN
padlib.obj	Funktionen PDnet_???	NEIN
padvdm.obj	Funktionen VDM_???	NEIN
makefile	Makedatei zum Compilieren des Beispielprogramms	NEIN
padtest	Ausführbares Beispielprogramm	JA

HINWEIS

In der Beschreibung wird nicht bei jeder Funktion, Variablen o.ä. angegeben, in welcher Bibliothek diese definiert ist, da eine logische Zuordnung über Ihren Namen bzw. der Funktionsgruppe ersichtlich ist.

4.9 DEC AlphaStation OpenVMS/AXP 6.2/7.0

Die Bibliothek wird in Form von Objektdateien (*.OBJ) geliefert. Diese Dateien werden von DEC C in das Programm eingebunden. Damit der C-Compiler die Dateien einbinden kann, müssen Sie die entsprechenden Dateien von dem Archiv auf der Diskette auf der Festplatte installieren und in das Objektdateiverzeichnis Ihres Projektes kopieren.

Folgende Dateien befinden sich im Archiv PADIT017.A auf der mitgelieferten Diskette:

Verzeichnis\Dateiname	Beschreibung	Weitergabe
SYS$LIBRARY:PADLIB.OLB	Bibliothek für Gerätetreiber	NEIN
SYS$LIBRARY:PDNET$DEF.H	Headerdatei für Gerätetreiber	NEIN
SYS$LIBRARY:PADDEF.H	Headerdatei für Gerätetreiber	NEIN
SYS$SYSROOT:[SYSMGR.PADITF] README.TXT	Änderungen der gedruckten Dokumentation	NEIN
SYS$SYSROOT:[SYSMGR.PADITF] PADDEFI.H	Headerdatei für Basis Typdefinitionen	NEIN
SYS$SYSROOT:[SYSMGR.PADITF] PADITF.H	Headerdatei für Funktionen PAD_???	NEIN
SYS$SYSROOT:[SYSMGR.PADITF] PADLIB.H	Headerdatei für Funktionen PDnet_???	NEIN
SYS$SYSROOT:[SYSMGR.PADITF] PADVDM.H	Headerdatei für Funktionen VDM_???	NEIN
SYS$SYSROOT:[SYSMGR.PADITF] PADTEST.H	Headerdatei für Beispielprogramm	NEIN
SYS$SYSROOT:[SYSMGR.PADITF] PADTEST.C	Quellcode für Beispielprogramm - 1. Teil	NEIN
SYS$SYSROOT:[SYSMGR.PADITF] PADTEST1.C	Quellcode für Beispielprogramm - 2. Teil	NEIN
SYS$SYSROOT:[SYSMGR.PADITF] PADTEST2.C	Quellcode für Beispielprogramm - 3. Teil	NEIN
SYS$SYSROOT:[SYSMGR.PADITF] PADITF.OBJ	Funktionen PAD_???	NEIN
SYS$SYSROOT:[SYSMGR.PADITF] PADLIB.OBJ	Funktionen PDnet_???	NEIN
SYS$SYSROOT:[SYSMGR.PADITF] PADVDM.OBJ	Funktionen VDM_???	NEIN
SYS$SYSROOT:[SYSMGR.PADITF] MAKEFILE.COM	Kommandodatei zum Compilieren des Beispielprogramms	NEIN
SYS$SYSROOT:[SYSMGR.PADITF] PADTEST.EXE	Ausführbares Beispielprogramm	JA

HINWEIS

In der Beschreibung wird nicht bei jeder Funktion, Variablen o.ä. angegeben, in welcher Bibliothek diese definiert ist, da eine logi-

sche Zuordnung über Ihren Namen bzw. der Funktionsgruppe ersichtlich ist.

4.9.1 DEC C

Die PAD-Interface Bibliothek wird wie folgt installiert:

- Legen Sie die Diskette mit der Datei PADIT017.A in das Diskettenlaufwerk.
- Starten Sie das Betriebsystem OpenVMS.
- Melden Sie sich mit System Account an (Benutzer SYSTEM).
- Öffnen Sie ein DECterm Fenster.
- Rufen Sie VMSINSTAL auf. Als Parameter werden der Archivname und der Device Name des Diskettenlaufwerk angegeben:

```
$ @sys$update:vmsinstal padit017 dva0:
```

Die Installation kann jederzeit mit Ctrl+Y abgebrochen werden.

- Beantworten Sie die von der Installation gestellten Fragen. Die Vorgaben in eckigen Klammern [] können mit der Return Taste übernommen werden.
- Logen Sie sich nach der Installation aus, da VMSINSTAL die Symboltabellen löscht und neu überschreibt:

```
$logout
```

Nach der Installation befindet sich die PAD-Interface Bibliothek im Verzeichnis SYS$SYSROOT:[SYSMGR.PADITF].

4.10 SUN IPC/IPX (SUN-OS 4.1.2)

Die PAD-Interface Bibliothek für SUN IPC/IPX (SUN-OS 4.1.2) Programme ist für folgende Compiler verfügbar.

4.10.1 GNU-C

Die Bibliothek wird in Form von Objektdateien (*.o) geliefert. Diese Dateien werden von GNU-C (gcc) in das Programm eingebunden. Damit der C-Compiler die Dateien einbinden kann, müssen Sie die entsprechenden Dateien von der Diskette in das Verzeichnis Ihres Projektes kopieren. GNU-C ist erforderlich, da die Bibliothek in ANSI-C geschrieben ist. Der Compiler cc ist kein ANSI-C-Compiler und hat nicht das IEEE-Format für float-Variablen.

Folgende Dateien befinden sich im Root-Verzeichnis auf der mitgelieferten Diskette:

Dateiname	Beschreibung	Weitergabe
readme.txt	Änderungen und Ergänzungen der gedruckten Dokumentation	NEIN
paddefi.h	Headerdatei für Basis Typdefinitionen	NEIN
paditf.h	Headerdatei für Funktionen PAD_???	NEIN
padlib.h	Headerdatei für Funktionen PDnet_???	NEIN
padvdm.h	Headerdatei für Funktionen VDM_???	NEIN
padtest.h	Headerdatei für Beispielprogramm	NEIN
padtest.c	Quellcode für Beispielprogramm - 1. Teil	NEIN
padtest1.c	Quellcode für Beispielprogramm - 2. Teil	NEIN
padtest2.c	Quellcode für Beispielprogramm - 3. Teil	NEIN
padapisu.o	Interface für SUN	NEIN
paditf.o	Funktionen PAD_???	NEIN
padlib.o	Funktionen PDnet_???	NEIN
padvdm.o	Funktionen VDM_???	NEIN
makefile	Batch-Datei zum Compilieren des Beispielprogramms	NEIN
padtest	Ausführbares Beispielprogramm	JA

HINWEIS

In der Beschreibung wird nicht bei jeder Funktion, Variablen o.ä. angegeben, in welcher Bibliothek diese definiert ist, da eine logi-

sche Zuordnung über Ihren Namen bzw. der Funktionsgruppe ersichtlich ist.

4.11 SUN Ultra 5 (Solaris 2.6)

Die PAD-Interface Bibliothek für SUN Ultra 5 (Solaris 2.6) Programme ist für folgende Compiler verfügbar.

4.11.1 SunPro C 4.2

Die Bibliothek wird in Form von Objektdateien (*.o) geliefert. Diese Dateien werden von SunPro C (cc) in das Programm eingebunden. Damit der C-Compiler die Dateien einbinden kann, müssen Sie die entsprechenden Dateien von der Diskette in das Verzeichnis Ihres Projektes kopieren.

Folgende Dateien befinden sich im Root-Verzeichnis auf der mitgelieferten Diskette:

Dateiname	Beschreibung	Weitergabe
readme.txt	Änderungen und Ergänzungen der gedruckten Dokumentation	NEIN
paddefi.h	Headerdatei für Basis Typdefinitionen	NEIN
paditf.h	Headerdatei für Funktionen PAD_???	NEIN
padlib.h	Headerdatei für Funktionen PDnet_???	NEIN
padvdm.h	Headerdatei für Funktionen VDM_???	NEIN
padtest.h	Headerdatei für Beispielprogramm	NEIN
padtest.c	Quellcode für Beispielprogramm - 1. Teil	NEIN
padtest1.c	Quellcode für Beispielprogramm - 2. Teil	NEIN
padtest2.c	Quellcode für Beispielprogramm - 3. Teil	NEIN
Padapis5.o	Interface für SUN Ultra 5	NEIN
paditf.o	Funktionen PAD_???	NEIN
padlib.o	Funktionen PDnet_???	NEIN
padvdm.o	Funktionen VDM_???	NEIN
makefile	Batch-Datei zum Compilieren des Beispielprogramms	NEIN
padtest	Ausführbares Beispielprogramm	JA

HINWEIS

In der Beschreibung wird nicht bei jeder Funktion, Variablen o.ä. angegeben, in welcher Bibliothek diese definiert ist, da eine logi-

sche Zuordnung über Ihren Namen bzw. der Funktionsgruppe ersichtlich ist.

4.12 Motorola-8420 (System V/m88k)

Die PAD-Interface Bibliothek für Motorola-8420 (System V/m88k) Programme ist für folgende Compiler verfügbar.

4.12.1 GNU-C

Die Bibliothek wird in Form von Objektdateien (*.o) geliefert. Diese Dateien werden von GNU-C (gcc) in das Programm eingebunden. Damit der C-Compiler die Dateien einbinden kann, müssen Sie die entsprechenden Dateien von der Diskette in das Verzeichnis Ihres Projektes kopieren. GNU-C ist erforderlich, da die Bibliothek in ANSI-C geschrieben ist. Der Compiler cc ist kein ANSI-C-Compiler und hat nicht das IEEE-Format für float-Variablen.

Folgende Dateien befinden sich im Root-Verzeichnis auf der mitgelieferten Diskette:

Dateiname	Beschreibung	Weitergabe
readme.txt	Änderungen und Ergänzungen der gedruckten Dokumentation	NEIN
paditf.h	Headerdatei für Funktionen PAD_???	NEIN
padlib.h	Headerdatei für Funktionen PDnet_???	NEIN
padvdm.h	Headerdatei für Funktionen VDM_???	NEIN
padtest.h	Headerdatei für Beispielprogramm	NEIN
padtest.c	Quellcode für Beispielprogramm - 1. Teil	NEIN
padtest2.c	Quellcode für Beispielprogramm - 2. Teil	NEIN
paditf.o	Funktionen PAD_???	NEIN
padlib.o	Funktionen PDnet_???	NEIN
padvdm.o	Funktionen VDM_???	NEIN
makefile	Batch-Datei zum Compilieren des Beispielprogramms	NEIN
padtest	Ausführbares Beispielprogramm	JA

HINWEIS

Damit der GNU C-Compiler unterscheiden kann, ob die Bibliothek auf einer SUN- oder Motorola-Workstation läuft, muß bei der Motorola-Workstation das Symbol m88k definiert sein. Dieses Symbol kann der GNU C-Compiler mit dem Argument -Dm88k erzeugen.

In der Beschreibung wird nicht bei jeder Funktion, Variablen o.ä. angegeben, in welcher Bibliothek diese definiert ist, da eine logische Zuordnung über Ihren Namen bzw. der Funktionsgruppe ersichtlich ist.

5 Typen Referenz

Dieses Kapitel beschreibt die in der PAD-Interface Bibliothek definierten Typen. Folgende Typen werden verwendet.

Typ	Beschreibung
Allgemeine Typen	
INT8 .. UINT32	Einheitliche Datentypen für Zahlen auf allen Plattformen
tPadFuncParams	Parameter der Funktion, die einen Fehler erzeugt hat.
Typen zum Verwalten des lokalen PADs	
tPAD_VersionInfo	Firmwareversion des lokalen PAD
Typen zum Stellen und Lesen der PDnet Uhrzeit	
tPadDate	Datum der PAD Echzeituhr
tPadTime	Zeit der PAD Echzeituhr
Typen zum Senden und Empfangen von Telegrammen	
tTelgram	Telegramm in Telegramm Funktionen
Typen der PDnet Lifeliste	
TstationIDs	Enthält eine Liste mit Stationsadressen
tExtLifeListTln	Informationen der erweiterten Lifeliste eines PADs
Typen zur Bearbeitung von VDM-Datenzellen	
tVdmHeader	Kopf einer VDM-Datenzelle
PVVAL	Prozeßwert in VDM Funktionen

5.1 INT8 .. UINT32

Für die Bibliothek existieren neue Datentypen, die auf allen Plattformen das gleiche Format haben.

Datentyp	Länge	Bereich	Pascal-Typ	C-Typ
INT8	8 Bits	-128 bis 127	shortint	signed char
UINT8	8 Bits	0 bis 255	byte	unsigned char
INT16	16 Bits	-32768 bis 32767	integer	signed short int
UINT16	16 Bits	0 bis 65535	word	unsigned short int
INT32	32 Bits	-2147483648 bis 2147483647	longint	signed long int
UINT32	32 Bits	0 bis 4294967295	longint	unsigned long int

Für die Pascal Bibliothek sind noch weitere Datentypen definiert, die Zeiger auf die oben genannten Datentypen bilden.

Datentyp	Länge	Beschreibung	Pascal-Typ
pINT8	32 Bits	Zeiger auf INT8	^INT8
pUINT8	32 Bits	Zeiger auf UINT8	^UINT8
pINT16	32 Bits	Zeiger auf INT16	^INT16
pUINT16	32 Bits	Zeiger auf UINT16	^UINT16
pINT32	32 Bits	Zeiger auf INT32	^INT16
pUINT32	32 Bits	Zeiger auf UINT32	^UINT16

5.2 PVVAL

AUFGABE

Die Datenstruktur PVVAL enthält einen Prozeßwert in der Strukturvariablen mit dem zugehörigen VDM-Datentyp.

PASCAL DEFINITION

```
type PVVAL =
  record
    case byte of
      0: (pv_byte:       UINT8);
      1: (pv_shortint:   INT8);
      2: (pv_word:       UINT16);
      3: (pv_integer:    INT16);
      4: (pv_doubleword: UINT32);
      5: (pv_longint:    INT32);
      6: (pv_single:     single);
      7: (pv_double:     double);
      8: (pv_extended:   extended);
      9: (pv_ascii:
         array[0..C_PV_STRING_LEN] of char);
     10: (pv_string:
                 string[C_PV_STRING_LEN+1]);
     11: (pv_record:
         array[0..C_PV_STRING_LEN] of char);
     12: (pv_bit:        UINT8);
     13: (pv_bit8:       UINT8);
     14: (pv_bit16:      UINT16);
     15: (pv_bit32:      UINT32);
   end;
```

C DEFINITION

```
typedef union
{
  UINT8  pv_byte;
  INT8   pv_shortint;
  UINT16 pv_word;
  INT16  pv_integer;
  UINT32 pv_doubleword;
  INT32  pv_longint;
  float  pv_single;
  double pv_double;
  long double pv_extended;
  char   pv_ascii[C_PV_STRING_LEN+1];
  char   pv_string[C_PV_STRING_LEN+2];
  char   pv_record[C_PV_STRING_LEN+1];
  UINT8  pv_bit;
  UINT8  pv_bit8;
  UINT16 pv_bit16;
  UINT32 pv_bit32;
} PVVAL;
```

BESCHREIBUNG

Im Typ PVVAL wird ein Prozeßwert abgelegt. Je nach Datentyp muß das zugehörige Feld gelesen bzw. beschrieben werden.

SIEHE AUCH

VDM_Read_PV, VDM_Write_PV

5.3 tExtLifeListTln

AUFGABE

Das Array tExtLifeListTln wird von der Funktion PDnet_ExtLifeList zurückgegeben und enthält für einen PDnet-Controller die parametrierten Treiber und deren Fehlermeldungen.

PASCAL DEFINITION

```
tExtLifeListEntry =
  record
    ID:    UINT8;
    Flags: UINT8;
  end;
pExtLifeListEntry = ^tExtLifeListEntry;

tExtLifeListTln
  = array[0..15] of tExtLifeListEntry;
pExtLifeListTln = ^tExtLifeListTln;
```

C DEFINITION

```
typedef
  struct
  {
    UINT8 ID;
    UINT8 Flags;
  } tExtLifeListEntry;

typedef tExtLifeListEntry tExtLifeListTln[16];
typedef tExtLifeListTln _*pExtLifeListTln;
```

FELDER

- ID
  In ID steht die Nummer CTR_??? des parametrierten Treibers.
- Flags
  In Flags steht die Bitmaske CTRFL_??? mit den Statusflags des zugehörigen Treibers.

BESCHREIBUNG

Das Array tExtLifeListTln 16 Elemente (0..15) von der Struktur tExtLifeListEntry. Die Struktur besteht aus den beiden Bytewerten ID und Flags. In ID steht die Nummer CTR_??? des parametrierten Treibers und in Flags die Bitmaske CTRFL_??? mit den dem Treiber zugehörigen Statusflags.

Eine Ausnahme gibt es beim Array Element 0: In ExtLifeListTln[0].ID steht der Localstatus des Teilnehmers. Über die enthaltenen CTRFL_LOCAL_LAN_??? Flags wird angezeigt, ob der Teilnehmer über LAN-A oder LAN-B des lokalen PDnet-Controllers

erreichbar ist. In ExtLifeListTln[0].Flags steht die Gruppenadresse des abgefragten Teilnehmers.

ACHTUNG

Bei der Firmwareversion 1.x wird keine erweiterte Life-Liste unterstützt. Aus Kompatibilitätsgründen liefert die Funktion den Localstatus und die Gruppenadresse in ExtLifeListTln[0].

Da die ersten Firmwareversionen 2.x noch nicht alle Treibernummern anzeigen, aber die Statusflags der Treiber, sind in dem Array ExtLifeListEntry die Indizes 1 bis 6 einem bestimmten Treiber zugeordnet, wenn die ID = CTR_NULL ist. Folgende Treiber stehen in den Indizes 1..6 von ExtLifeListTln[], wenn die Treibernummer ExtLifeListTln[1..6].ID fehlt:

Index	Treiber
1	Modul Kernel
2	Modul Hardwaretest
3	Modul Setupdaten
4	LAN Treiber
5	Hostinterface
6	Serieller Kanal 1

Die Treiber "Hostinterface" und "Serieller Kanal" können mehr als einmal auftreten und werden dann ab Index 7 heraufgezählt. Ist z. B. ExtLifeListTln[6].ID = CTR_NULL und ExtLifeListTln[7].ID enthält die Treibernummer eines Seriellen Kanals, so ist der "Serielle Kanal 1" nicht belegt (Index 6) und der "Serielle Kanal 2" ist belegt (Index 7).

SIEHE AUCH

CTR_???, CTRFL_???, PDnet_ExtLifeList

5.4 tPAD_VersionInfo

AUFGABE

Enthält die Firmwareversion, die von der Funktion PAD_VersionInfo geliefert wird.

PASCAL DEFINITION

```
tPAD_VersionInfo =
  record
    len_lo:    UINT8;
    len_hi:    UINT8;
    jmp:       array[0..1]  of UINT8;
    cr:        array[0..12] of UINT8;
    FwStatus:  UINT8;
    FwVersion: UINT8;
    FwRelease: UINT8;
    FwDay:     UINT8;
    FwMonth:   UINT8;
    FwYear:    UINT8;
  end;
```

C DEFINITION

```
typedef struct {
  UINT8  len_lo;
  UINT8  len_hi;
  UINT8  jmp[2];
  UINT8  cr[13];
  UINT8  FwStatus;
  UINT8  FwVersion;
  UINT8  FwRelease;
  UINT8  FwDay;
  UINT8  FwMonth;
  UINT8  FwYear;
} tPAD_VersionInfo;
```

FELDER

- FwVersion
  Enthält die Versionsnummer der Firmware im BCD-Format.
- FwRelease
  Enthält die Releasenummer. der Firmware im BCD-Format.
- FwDay
  Enthält den Tag der Firmware im BCD-Format.
- FwMonth
  Enthält den Monat der Firmware im BCD-Format
- FwYear
  Enthält das Jahr der Firmware im BCD-Format.

Die restlichen Fehler werden nur intern verwendet.

SIEHE AUCH

PAD_VersionInfo

5.5 tPadDate

AUFGABE

Die Struktur tPadDate enthält das Datum der PAD Echzeituhr.

PASCAL DEFINITION

```
type tPadDate =
  record
    da_year: INT16;
    da_day:  INT8;
    da_mon:  INT8;
  end;
```

C DEFINITION

```
typedef struct {
  INT16 da_year;
  INT8  da_day;
  INT8  da_mon;
} tPadDate;
```

FELDER

- da_year
  Enthält das Jahr des Datums.
- da_day
  Enthält den Tag des Datums.
- da_mon
  Enthält den Monat des Datums.

SIEHE AUCH

PAD_GetDateTime, PAD_SetDateTime

5.6 tPadFuncParams

AUFGABE

Liefert Parameter der Funktion, die einen Fehler erzeugt hat.

PASCAL DEFINITION

```
type tPadFuncParams =
record
  FuncNr:     UINT16;
  FuncReturn: INT16;
  FuncParam:  array[0..6] of UINT32;
end;
```

C DEFINITION

```
typedef struct
{
  UINT16 FuncNr;
  INT16  FuncReturn;
  UINT32 FuncParam[7];
} tPadFuncParams;
```

FELDER

- FuncNr
  Das Feld FuncNr enthält eine CFUNCNR_??? Konstante der Funktion, die den letzten Fehler verursacht hat.
- FuncReturn
  Das Feld FuncReturn enthält den Rückgabewert der Funktion, die den letzten Fehler verursacht hat.
- FuncParam
  Das Array FuncParam enthält die Parameter der Funktion, die den letzten Fehler verursacht hat.

BESCHREIBUNG

Wenn eine Funktion der PAD-Interface Bibliothek einen Fehler erkennt, liefert sie einen CPDNET_??? oder CVDM_??? Fehlercode zurück. Gleichzeitig wird der zuletzt aufgetretene Fehlercode in einer globalen Variablen abgelegt, die mit PAD_Result gelesen werden kann. Da einige Bibliotheksfunktionen auch andere Bibliotheksfunktionen aufrufen, kann nicht genau ermittelt werden, welche Bibliotheksfunktion den Fehlercode aufgrund welcher Parameter ausgelöst hat. Deshalb wurden zwei globale Strukturen in die Bibliothek aufgenommen, welche bei einem Fehlercode zusätzliche Informationen über die aktuelle Funktion und evt. über die bearbeitete VDM-Datenzelle speichern.

Die Strukturen sind in der PAD-Interface Bibliothek enthalten, wenn in ihr das Symbol DEBUG_PARAMS definiert ist. Beim Auftreten eines Fehlercodes wird in der Struktur PadFuncParams die Funktionsnummer CFUNCNR_???, der Rückgabewert der Funktion und die Funktionsparameter abgelegt.

5.7 TStationIDs

AUFGABE

Enthält eine Liste mit Stationsadressen.

PASCAL DEFINITION

```
type TStationIDs : array[0..255] of UINT8;
```

C DEFINITION

```
typedef UINT8 TStationIDs[256];
```

BESCHREIBUNG

Die Liste enthält mehrere Stationsadressen und wird mit einer 0 abgeschlossen.

SIEHE AUCH

PDnet_GroupMemebers

5.8 tTelegram

AUFGABE

Enthält ein zu sendendes bzw. ein empfangenes Telegramm.

PASCAL DEFINITION

```
type tTelegram =
  record
    Semaphore: UINT8;
    PDnetId:   UINT8;
    GroupId:   UINT8;
    Len:       UINT16;
    Typ:       UINT8;
    Data:      array[0..504] of UINT8;
  end;
pTelegram = ^tTelegram;
```

C DEFINITION

```
typedef struct
{
  UINT8  Semaphore;
  UINT8  PDnetId;
  UINT8  GroupId;
  UINT16 Len;
  UINT8  Typ;
  UINT8  Data[505];
} tTelegram;
typedef tTelegram *pTelegram;
```

FELDER

- Semaphore
  Die Semaphore kennzeichnet einen Telegrammpuffer als belegt, wenn sie ungleich 0 ist. Ist die Semaphore 0, enthält der Puffer kein Telegramm.
- PDnetId
  In PDnetId steht die PDnet Stationsadresse der Zielstation, zu dem das Telegramm gesendet werden soll bzw. die Quellstation, von der das Telegramm empfangen wurde. Ist PDnetId 0, wird ein Broadcast-Telegramm an alle Stationen gesendet. Wenn in PDnetId die eigene PDnet Stationsadresse eingetragen wird, verarbeitet der lokale PAD das Telegramm nur intern.
- GroupId
  Bei zu sendenden Telegrammen enthält GroupId die PDnet Gruppenadresse an die das Telegramm gesendet werden soll. Ist GroupId 0, wird ein direkt adressiertes Telegramm gesendet. Wenn die GroupId größer als 0 ist, wird die PDnet Stationsadresse ignoriert und das Telegramm an alle Gruppenteilnehmer gesendet die zu diesem Zeitpunkt online sind (Gruppenadressierung).
- Len
  Das Feld Len enthält die Telegrammlänge in Byte. Sie berechnet sich aus der Größe der in Data[] eingetragenen Daten zuzüglich ein Byte für den Telegrammtyp.
- Typ
  Das Feld Typ enthält den Telegrammtyp. Einige Telegrammtypen sind vom PDnet fest vergeben und dürfen nicht anderweitig verwendet werden.
- Data
  Das Array Data enthält die Daten des Telegramms. Der Inhalt ist abhängig vom Telegrammtyp.

BESCHREIBUNG

Ein Telegramm ist ein Datenpaket, welches im PDnet von einer Station zu einer anderen Station (direkt adressiertes Telegramm) oder mehreren Stationen (Gruppenadressierung) übertragen wird. Ein Telegramm besteht aus einem Header und einem Datenteil. Der Header bestimmt, wie der Datenteil zu interpretieren ist.

SIEHE AUCH

PAD_ReceiveTelegram, PAD_SendTelegram

5.9 tPadTime

AUFGABE

Die Struktur tPadTime enthält die Uhrzeit der PAD Echzeituhr.

PASCAL DEFINITION

```
type tPadTime =
  record
    ti_min:  UINT8;
    ti_hour: UINT8;
    ti_hund: UINT8;
    ti_sec:  UINT8;
  end;
```

C DEFINITION

```
typedef struct {
  UINT8  ti_min;
  UINT8  ti_hour;
  UINT8  ti_hund;
  UINT8  ti_sec;
} tPadTime;
```

FELDER

- ti_min
  Enthält die Minuten der Uhrzeit.
- ti_hour
  Enthält die Stunden der Uhrzeit.
- ti_hund
  Enthält die Hundertstel Sekunden der Uhrzeit
- ti_sec
  Enthält die Sekunden der Uhrzeit

SIEHE AUCH

PAD_GetDateTime, PAD_SetDateTime

5.10 tVdmHeader

AUFGABE

Der Typ tVdmHeader enthält den Kopf einer VDM-Datenzelle.

PASCAL DEFINITION

```
type tVdmHeader =
  record
    Flags1:     UINT8;
    DataCellID: UINT16;
    DataType:   UINT8;
    DataSize:   UINT8;
    DataCount:  UINT16;
    AddrPos:    UINT16;
    AddrKind:   UINT8;
    PDnetId:    UINT16;
    NextCell:   UINT32;
end;
```

C DEFINITION

```
typedef struct {
  UINT8  Flags1;
  UINT16 DataCellID;
  UINT8  DataType;
  UINT8  DataSize;
  UINT16 DataCount;
  UINT16 AddrPos;
  UINT8  AddrKind;
  UINT16 PDnetId;
  UINT32 NextCell;
} tVdmHeader;
```

FELDER

- Flags1
  Im Feld Flags1 stehen interne CZF_??? Steuerungsbits der VDM-Datenzelle. Das Bit CZF_TRANSMITCELL legt fest, ob es sich um eine Sendezelle oder eine Empfangszelle handelt.
- DataCellId
  Das Feld DataCellId enthält die Kennung der VDM-Datenzelle. Sie wird nur intern im PDnet zur Adressierung der VDM-Datenzellen benutzt.
- DataType
  Im Feld DataType steht der CDT_??? Datentyp der folgenden Prozeßwerte.
- DataSize
  Das Feld DataSize, enthält die Größe der folgenden Prozeßwerte in Byte.
- DataCount
  Das Feld DataCount enthält die Anzahl der folgenden Prozeßwerte.

- AddrPos
  Das Feld AddrPos beinhaltet die PV-Nummer der des ersten Prozeßwertes der Datenzelle. Bei den folgenden Prozeßwerten wird die PV-Nummer erhöht.
- AddrKind
  Das Feld AddrKind legt die Adressierungsart der VDM-Datenzelle fest. Bei 0 wird eine direkte Adressierung verwendet. Bei 1 wird eine Gruppenadressierung verwendet.
- PDnetId
  Das Feld PDnetId enthält die Stationsadresse (bei direkter Adressierung) oder Gruppenadresse (bei Gruppenadressierung) der Gegenstation der Datenzelle.
- NextCell
  Im Feld NextCell steht die PAD-Speicheradresse der folgenden VDM-Datenzelle. Ist sie 0, folgen keine weiteren Datenzellen.

BESCHREIBUNG

Die VDM-Datenzellen werden im PAD als verkettete Liste abgelegt. Jede Datenzelle beginnt mit einem einheitlichen Datenzellenkopf tVdmHeader. In ihm stehen die globalen Informationen der Datenzelle wie z.B. der CDT_??? Datentyp der folgenden Prozeßwerte.

Hinter dem Datenzellenkopf folgen mehrere gleich große Elemente die aus einem Flagbyte und dem Prozeßwert bestehen. Das Flagbyte enthält CFB_??? Statusbits die angeben, ob der Prozeßwert gültig oder verändert ist.

SIEHE AUCH

CDT_???, CFB_???, VDM_GetNextVdmHeader

5.11 tVdmHeaderParams

AUFGABE

Liefert Informationen zur VDM-Datenzelle, bei der ein Fehler aufgetreten ist.

PASCAL DEFINITION

```
type tVdmHeaderParams =
record
  pHeader:    UINT32;
  PadPage:    UINT8;
  Flags1:     UINT8;
  DataCellID: UINT16;
  DataType:   UINT8;
  DataSize:   UINT8;
  DataCount:  UINT16;
  AddrPos:    UINT16;
  AddrKind:   UINT8;
  PDnetId:    UINT16;
  NextCell:   UINT32;
end;
```

C DEFINITION

```
typedef struct {
  UINT32 pHeader;
  UINT8  PadPage;
  UINT8  Flags1;
  UINT16 DataCellID;
  UINT8  DataType;
  UINT8  DataSize;
  UINT16 DataCount;
  UINT16 AddrPos;
  UINT8  AddrKind;
  UINT16 PDnetId;
  UINT32 NextCell;
} tVdmHeaderParams;
```

BESCHREIBUNG

Tritt in einer Bibliotheksfunktion ein Fehlercode beim Zugriff auf eine VDM-Datenzelle auf, so wird in der Struktur VdmHeaderParams der Header der im PAD bearbeiteten VDM-Datenzelle abgelegt. Außerdem wird der verwendete Zeiger auf die VDM-Datenzelle und die selektierte Seite des PAD-PC gesichert.

6 Konstanten Referenz

Dieses Kapitel beschreibt die in der PAD-Interface Bibliothek definierten Konstanten. Folgende Konstanten werden verwendet.

Konstante	Beschreibung
Konstanten für Rückgabewerte der Funktionen	
CPDNET_???	Fehlercode von PAD_??? und PDnet_??? Funktionen
CVDM_???	Fehlercode von VDM_??? Funktionen
Konstanten für den lokalen PAD	
CLED_??	Statusbits für PAD-Status Leuchtdioden
Konstanten der erweiterten Lifeliste	
CTR_???	Treibernummern der erweiterten Lifeliste
CTRFL_???	Meldungsbits der erweiterten Lifeliste
Konstanten für VDM-Datenzellen	
CZF_???	Statusbits im Flagbyte des VDM-Datenzellenkopf
CDT_???	VDM Datentypen
CFB_???	Statusbits im Flagbyte der Prozeßwerte

6.1 CDT_???

AUFGABE

Die Konstanten CDT_??? Enthalten alle Datentypen, die vom VDM definiert werden.

Konstante	Datentyp	Byte	Wertebereich/Bemerkung
Integer-Typen			
CDT_BYTE	Byte	1	0 .. 255
CDT_SHORTINT	ShortInteger	1	-128 .. 127
CDT_WORD	Word	2	0 .. 65535
CDT_INTEGER	Integer	2	-32768 .. 32767
CDT_DOUBLEWORD	DoubleWord	4	0 .. 4294967295
CDT_LONGINT	LongInteger	4	-2147483648 .. 2147483647
Real-Typen			
CDT_SINGLE	Single	4	IEEE-Typ: ShortReal
CDT_DOUBLE	Double	8	IEEE--Typ: LongReal
CDT_EXTENDED	Extended	10	
Text-Typen			
CDT_ASCII	ASCII-Zeichen	n	ASCII-Zeichen mit der parametrierten Länge oder abschließenden Byte 00h.
CDT_STRING	StringArray	n	Jeder String hat ein führendes Längenbyte und am Ende ein Byte 00h, so daß dieser Typ von den Hochsprachen PASCAL und C leicht zu verarbeiten ist. Beide Byte werden in der Längenangabe n berücksichtigt!
Komplexe Typen			
CDT_RECORD	Struktur	n	Mit diesem Typ können strukturierte Datentypen verarbeitet werden
Bitfeld Typen			
CDT_BIT	Bitfeld (Merker)	1	Merkerwert in Bit 0
CDT_BIT8	8-Bitfeld	1	8 Merker in einem Byte
CDT_BIT16	16-Bitfeld	2	16 Merker in einem Wort
CDT_BIT32	32-Bitfeld	4	32 Merker in einen Doppelwort

SIEHE AUCH

VDM_Read_PV, VDM_Write_PV

6.2 CFB_???

AUFGABE

Die Konstanten CFB_??? beschreiben die Bits des Flagbyte für jeden Prozeßwert einer VDM-Datenzelle.

KONSTANTEN

- CFB_DATE_CHANGED
  Dieses Bit zeigt an, daß der folgende Prozeßwert geändert wurde. Bei einer Sendezelle wird dieses Bit von der Bibliothek gesetzt, wenn ein geänderter Prozeßwert geschrieben wird.. Der VDM löscht das Bit, wenn er den Prozeßwert übertragen hat.

  Bei einer Empfangszelle setzt der VDM das Bit, wenn ein geänderter Prozeßwert empfangen wird. Die Bibliothek löscht das Bit, wenn der Prozeßwert gelesen wird.
- CFB_DATE_VALID
  Dieses Bit ist gesetzt, wenn der Prozeßwert gültig ist. Bei einer Sendezelle wird dieses Bit von der Bibliothek gesetzt, wenn ein Prozeßwert geschrieben wird.

  Bei einer Empfangszelle setzt der VDM das Bit, wenn ein Prozeßwert empfangen wird.

SIEHE AUCH

VDM_Read_PV, VDM_Write_PV

6.3 CLED_???

AUFGABE

Die Konstanten CLED_??? dienen zum Zugriff auf die Statusbits der PAD-Status Leuchtdioden.

KONSTANTEN

- CLED_PADSTATUS
  Leuchtdiode PAD-Status.
- CLED_HARDWARE
  Leuchtdiode Hardware
- CLED_SETUP
  Leuchtdiode Setup
- CLED_REMOTEPG
  Leuchtdiode RemotePG
- CLED_BUS
  Leuchtdiode Bus
- CLED_SERIEL1
  Leuchtdiode Seriell 1
- CLED_SERIEL2
  Leuchtdiode Seriell 2
- CLED_SMI
  Leuchtdiode SMI

SIEHE AUCH

PAD_GetStatusLeds

6.4 CPDNET_???

AUFGABE

Die Funktionen PAD_??? und PDnet_??? liefern als Status eine Konstante CPDNET_???, welche den Erfolg oder Fehlercode meldet.

KONSTANTEN

- CPDNET_OK
  Es ist kein Fehler aufgetreten.
- CPDNET_NO_INIT
  Es wurde keine Initialisierung der Bibliothek mit PAD_Init durchgeführt.
- CPDNET_NO_CONTROLLER_DETECTED
  Die Funktion PAD_Init konnte keinen PAD finden. Überprüfen Sie die Parameter von PAD_Init sowie die Hardwareeinstellungen des PAD.
- CPDNET_NO_REINIT_ALLOWED
  Ein zweiter Aufruf von PAD_Init mit den selben Ressourcen ist nicht gestattet. Deaktivieren Sie den PAD über PAD_Done um ihn wieder zu aktivieren.
- CPDNET_BAD_PARAM
  An eine Funktion wurden nicht zulässige Parameter übergeben (leere Zeiger, unzulässige Datennummer, usw.).
- CPDNET_RESTART_RUNNING
  Die Funktion wurde abgebrochen, weil sie einen laufenden PAD-Restart erkannt hat. Die nachfolgend aufgerufenen Funktionen liefern CPDNET_RESTART_RUNNING, bis der PAD-Restart beendet ist.
- CPDNET_RESTART_END
  Die Funktion wurde abgebrochen, weil sie das Ende eines PAD-Restart erkannt hat. Die nachfolgend aufgerufenen Funktionen können wieder auf den PAD zugreifen.
- CPDNET_TIMEOUT
  Die Funktion konnte wurde wegen eines Timeout Fehler abgebrochen.
- CPDNET_NO_SYNC
  Die Bibliothek konnte den Prozeß mit dem PAD nicht synchronisieren.
- CPDNET_TX_FULL
  Für das Telegramm ist kein freier Sendepuffer verfügbar.
- CPDNET_RX_FULL
  Im Empfangspuffer liegt ein neues Telegramm.

- CPDNET_RX_EMPTY
  Der Empfangspuffer für Telegramme ist leer.
- CPDNET_BAD_SEGM
  An die Funktion wurde eine ungültige Adresse übergeben.
- CPDNET_NOT_ENOUGH_MEMORY
  Auf dem Heap ist nicht mehr genug Speicher vorhanden, um die internen Datenstrukturen der Bibliothek anzulegen.
- CPDNET_ERR_NO_FREE_CHANNEL
  Es wurde versucht, zu viele lokale PADs zu öffnen.
- CPDNET_ERR_NO_RESOURCES
  Die gewählte Gerätenummer hat keine eingetragenen Ressourcen.
- CPDNET_ERR_PAD_ACCESS_DENIED
  Der PAD mit der übergebenen Gerätenummer ist bereits geöffnet.
- CPDNET_ERR_DEVICE_DRIVER_NOT_FOUND
  PAD_Init hat den Gerätetreiber nicht gefunden. Entweder wurde der Gerätetreiber noch nicht installiert oder es liegt ein Ressourcenkonflikt vor.
- CPDNET_ERR_GET_LDTSELECTOR
  PAD_Init hat vom Gerätetreiber keinen Zugriff auf den PAD-Speicher bekommen.

6.5 CTR_???

AUFGABE

Die erweiterte Lifeliste enthält im Feld ID eine der folgenden CTR_??? Treibernummern.

KONSTANTEN

- CTR_NULL
  Die Modul-ID fehlt.
- CTR_KERNEL
  Das Kernelmodul ist aktiv.
- CTR_HARDWARETEST
  Das Hardwaretest Modul ist aktiv.
- CTR_SETUPDATEN
  Das Setupdaten Modul ist geladen.
- CTR_PDNET
  Der PDnet Kanaltreiber ist aktiv.
- CTR_SMI_PADPC
  Der Endgerätetreiber SMI PAD-PC ist aktiv
- CTR_SMI_PADPG
  Der Endgerätetreiber SMI PAD-PG
- CTR_SMI_SBUS
  Der Endgerätetreiber SMI-SBUS ist aktiv.
- CTR_SMI_VME
  Der Endgerätetreiber SMI-VME ist aktiv.
- CTR_SMI_IPC
  Der Endgerätetreiber SMI-IPC ist aktiv.
- CTR_PAD5035
  Der Endgerätetreiber PAD-5035 ist aktiv.
- CTR_PAD250
  Der Endgerätetreiber PAD-250 ist aktiv.
- CTR_PAD120
  Der Endgerätetreiber PAD-120 ist aktiv.
- CTR_PAD120LC
  Der Endgerätetreiber PAD-120lc ist aktiv.
- CTR_PADS5A
  Der Endgerätetreiber PAD-S5 Adreßlücke ist aktiv.
- CTR_PADS5K
  Der Endgerätetreiber PAD-S5 Kacheltreiber ist aktiv.

KERNEL TREIBER

HARDWARE TREIBER

SETUPDATEN TREIBER

LAN TREIBER

HOSTINTERFACE TREIBER

- CTR_MICRO
  Der Endgerätetreiber PAD-120 Modicon-Micro ist aktiv.
- CTR_TMCU
  Der Endgerätetreiber PAD-TMCU ist aktiv.
- CTR_T200
  Der Endgerätetreiber PAD-200 ist aktiv.
- CTR_CS31
  Der Endgerätetreiber PAD-31 ist aktiv.
- CTR_PAD140
  Der Endgerätetreiber PAD-140, Quantum ist aktiv.
- CTR_MODNET2ND_PAD5035
  Der Endgerätetreiber PAD-5035, ModNet-2ND ist aktiv.
- CTR_MODNET2ND_PAD250
  Der Endgerätetreiber PAD-250, ModNet-2ND ist aktiv.
- CTR_BLOCK_PAD120
  Der Endgerätetreiber PAD-120 mit VDM-Blocktreiber ist aktiv.
- CTR_BLOCK_PAD250
  Der Endgerätetreiber PAD-250 mit VDM-Blocktreiber ist aktiv.
- CTR_BLOCK_PAD5035
  Der Endgerätetreiber PAD-5035 mit VDM-Blocktreiber ist aktiv.

SERIELLE TREIBER

- CTR_SEAB1_MASTER
  Der Kanaltreiber SEAB1 Master ist aktiv.
- CTR_SEAB1_SLAVE
  Der Kanaltreiber SEAB1 Slave ist aktiv.
- CTR_3964R
  Der Kanaltreiber 3964R Master+Slave ist aktiv.
- CTR_ASCII
  Der Kanaltreiber ASCII ist aktiv.
- CTR_DIN19244_MASTER
  Der Kanaltreiber DIN 19244 (SEAB-1W) Master ist aktiv.
- CTR_DIN19244_SLAVE
  Der Kanaltreiber DIN 19244 (SEAB-1W) Slave ist aktiv.
- CTR_MODBUS_MASTER
  Der Kanaltreiber ModBus Master ist aktiv.
- CTR_MODBUS_SLAVE
  Der Kanaltreiber ModBus Slave ist aktiv.
- CTR_MODEM_BUS
  Der Kanaltreiber ModemTokenBus ist aktiv.
- CTR_THYROM
  Der Kanaltreiber ThyroM ist aktiv.

- CTR_RP57X_MASTER
  Der Kanaltreiber RP570/71 Master ist aktiv.
- CTR_RP57X_SLAVE
  Der Kanaltreiber RP570/71 Slave ist aktiv.
- CTR_PDNET_RAS_SERVER
  Der Kanaltreiber PDnet RAS Server ist aktiv.
- CTR_PDNET_RAS_CLIENT
  Der Kanaltreiber PDnet RAS Client ist aktiv.
- CTR_AWD_SEAB1F
  Der Kanaltreiber SEAB1F-AWD ist aktiv.
- CTR_AG95_INTERFACE
  Der Kanaltreiber AG95 Interface ist aktiv.
- CTR_PGEMU_S5
  Der Kanaltreiber PG-Emulator S5 ist aktiv.
- CTR_PGITF_S5
  Der Kanaltreiber PG-Interface S5 ist aktiv.
- CTR_PGEMU_A120_A250
  Der Kanaltreiber PG-Emulator A120 / A250 ist aktiv.
- CTR_PGITF_A120_A250
  Der Kanaltreiber PG-Interface A120 / A250 ist aktiv.
- CTR_PGEMU_MICRO
  Der Kanaltreiber PG-Emulator Micro ist aktiv.
- CTR_PGITF_MICRO
  Der Kanaltreiber PG-Interface Micro ist aktiv.
- CTR_PGEMU_A350_A500
  Der Kanaltreiber PG-Emulator A350 / A500 ist aktiv.
- CTR_PGITF_A350_A500
  Der Kanaltreiber PG-Interface A350 / A500 ist aktiv.
- CTR_PGEMU_CS31
  Der Kanaltreiber PG-Emulator CS31 ist aktiv.
- CTR_PGITF_CS31
  Der Kanaltreiber PG-Interface CS31 ist aktiv.

VDM TREIBER

- CTR_VDM
  Der Virtuelle Datenmodellmanager ist aktiv.

AWD TREIBER

- CTR_AWD
  Der Automatische Wähldienst ist aktiv.

GLEICHLAUF TREIBER

- CTR_GL
  Die Gleichlaufüberwachung ist aktiv.

SIEHE AUCH

CTRFL_???, tExtLifeListTln, PDnet_ExtLifeList

6.6 CTRFL_???

AUFGABE

Die erweiterte Lifeliste enthält im Feld Flags die folgenden CTRFL_??? Bits als Statusmeldung. Jede Meldungsgruppe ist einem bestimmten Feld Flags zugeordnet das durch das Feld ID bestimmt wird.

KONSTANTEN

- CTRFL_LOCAL_LAN_A
  Der PAD ist über Kanal LAN-A erreichbar.
- CTRFL_LOCAL_LAN_B
  Der PAD ist über Kanal LAN-B erreichbar.

KERNEL MELDNGEN

- CTRFL_KERNEL_RESTART
  Durch den Watchdog wurde ein Restart des PAD ausgelöst.
- CTRFL_KERNEL_TASKLISTOVER
  Die Firmware konnte keinen weiteren Task starten, weil in der Taskliste kein Platz mehr frei ist.
- CTRFL_KERNEL_MEMORYALLOC
  Die Firmware hat keinen freien Speicher mehr, um weitere Datenstrukturen anzulegen.
- CTRFL_KERNEL_REALTIMECLOCK
  Die Echtzeituhr im PAD wurde noch nicht initialisiert.

HARDWARE MELDUNGEN

- CTRFL_HARDWARE_CMOSRAM
  Das CMOS-RAM ist fehlerhaft.
- CTRFL_HARDWARE_FPROM
  Das FlashProm ist fehlerhaft.
- CTRFL_HARDWARE_FPROMVOLT
  Das FlashProm hat keine Programmierspannung.
- CTRFL_HARDWARE_LAN_A
  Auf dem Kanal LAN-A sind Fehler aufgetreten.
- CTRFL_HARDWARE_LAN_B
  Auf dem Kanal LAN-B sind Fehler aufgetreten.

SETUPDATEN MELDUNGEN

- CTRFL_SETUP_NONE
  Es sind keine Setupdaten vorhanden. Mit NetPro müssen neue Setupdaten in den PAD programmiert werden.
- CTRFL_SETUP_CHECKSUM
  Die Prüfsumme der Setupdaten ist fehlerhaft. Mit NetPro müssen neue Setupdaten in den PAD programmiert werden.
- CTRFL_SETUP_ID
  Die Firmware kennt ein Teil der Setupdaten nicht. Mit NetPro

müssen neue Setupdaten in den PAD programmiert werden. Oder die Firmwareversion stimmt nicht mit der von NetPro unterstützen Firmwareversionen überein.

HOSTINTERFACE MELDUNGEN

- CTRFL_HOST_USERPROGRAM
  Auf dem Endgerät läuft kein Anwenderprogramm, das den Software Watchdog im PAD nachtriggert.
- CTRFL_HOST_ERRORFLAG
  Der Endgerätetreiber hat einen Fehler im Endgerät gemeldet.
- CTRFL_HOST_BATTERY
  Die Pufferbatterie im Endgerät ist leer.
- CTRFL_HOST_TXTELEGRAM
  Ein gesendetes Telegramm hat ein fehlerhaftes Format.
- CTRFL_HOST_RXTELEGRAM
  Ein empfangenes Telegramm hat das vorher empfangene Telegramm überschrieben, bevor es ausgelesen wurde.

SERIELLE MELDUNEGN

- CTRFL_SERIAL_FRAMEERROR
  Im seriellen Protokoll sind Übertragungsfehler aufgetreten.
- CTRFL_SERIAL_TIMEOUT
  Ein serielles Datenpaket oder eine Quittierung ist nicht in der vom Protokoll definierten Zeitspanne angekommen.
- CTRFL_SERIAL_STATIONERROR
  Von einem Endgerät wurde ein Fehler gemeldet.

SIEHE AUCH

CTR_???, tExtLifeListTln, PDnet_ExtLifeList

6.7 CVDM_???

AUFGABE

Die Funktionen VDM_??? liefern als Status eine Konstante, welche den Erfolg oder Fehlercode meldet. Der Status entspricht einer Konstanten CPDNET_???, wenn ein Fehler in einer aufgerufenen Funktion PAD_??? auftrat, oder eine Konstante CVDM_???, wenn der Fehler in einer Funktion VDM_??? auftrat.

KONSTANTEN

- CVDM_OK
  Es ist kein Fehler aufgetreten.
- CVDM_NO_SYNC
  Eine VDM-Datenzelle konnte nicht gesperrt werden. Sie wird erst beim nächsten Durchlauf bearbeitet.
- CVDM_DATE_IDX
  Eine angegebene PV-Nummer ist nicht im VDM-Datenmodell des lokalen PADs vorhanden.
- CVDM_BAD_PARAM
  Falscher Parameter (z.B. Nullzeiger).
- CVDM_NOT_FOUND
  Die VDM-Datenzelle(n) wurde nicht gefunden.
- CVDM_DATACELL_WRONG
  Es wurden Unterschiede zwischen einer VDM-Datenzelle und der Kopie des Datenzellenkopfes entdeckt.
- CVDM_BAD_DATATYPE
  Der übergebene VDM-Datentyp stimmt nicht überein mit dem Typ der VDM-Datenzelle.
- CVDM_BAD_ADDRPOS
  Die übergebene Adresse liegt nicht in der VDM-Datenzelle.
- CVDM_BAD_DATACOUNT
  Die Übergebene Anzahl der PV's paßt nicht in die VDM-Datenzelle.
- CVDM_BAD_DATASIZE
  In der VDM-Datenzelle stimmt die eingetragene Größe des Datentyps nicht mit der zugehörigen Datentypgröße überein.
- CVDM_NO_CONNECTION
  Die Empfangsdatenzelle hat keine Verbindung zur Quellstation.
- CVDM_UNKNOWN_DATATYPE
  Unbekannter VDM-Datentyp.
- CVDM_NO_TRANSMITCELL
  PAD_Write_PV kann nicht in Empfangsdatenzellen schreiben.

SIEHE AUCH

VDM_Read_PV, VDM_Write_PV, VDM_Check_PV

6.8 CZF_???

AUFGABE

Die Konstanten CZF_??? Beschreiben die Bits im Feld Flag1 des VDM-Datenzellenkopfes.

KONSTANTEN

- CZF_BUSY
  Dieses Bit zeigt an, daß eine Datenübertragung für diese Zelle läuft, die auf mehrere Telegramme verteilt ist. Solange dieses Bit gesetzt ist, kann der Inhalt einer Empfangszelle inkonsistent sein.
- CZF_TRANSMITCELL
  Wenn dieses Bit gesetzt ist, handelt es sich bei dieser Datenzelle um eine Sendezelle, andernfalls um eine Empfangszelle.
- CZF_SENDALL
  Ist dieses Bit gesetzt (Alles Senden: Ein), werden beim Einschalten eines PDnet-Controllers und beim Ändern eines Prozeßwertes alle Prozeßwerte der VDM-Datenzelle neu übertragen. Bei VDM-Datenzellen mit Meldungen von der SPS zum Leitsystem sollte im NetPro „Alles Senden: Ein“ gesetzt werden, damit alle Änderungen der SPS im Leitsystem angezeigt werden.

  Ist dieses Bit zurückgesetzt (Alles Senden: Aus), werden beim Einschalten des PDnet-Controllers die Prozeßwerte nicht übertragen. Erst wenn sich ein Prozeßwert ändert, wird dieser allein übertragen. Bei VDM-Datenzellen mit Befehlen von dem Leitsystem zur SPS sollte im NetPro „Alles Senden: Aus“ gesetzt werden, damit nur geänderte Befehle an die SPS gesendet werden.
- CZF_CONNECTIONTOSOURCE
  Über dieses Bit wird dem Endgerät mitgeteilt, daß zu seiner Empfangszelle eine einwandfreie Verbindung von der zugehörigen Sendezelle besteht.
- CZF_CHANGED
  Dieses Bit zeigt an, daß mindestens ein Prozeßwert der VDM-Datenzelle geändert wurde. Bei Sendezellen wird dieses Bit von der Bibliothek gesetzt. Wenn der VDM alle geänderten Prozeßwerte abgearbeitet hat, setzt er dieses Bit auf 0 zurück.

  Bei Empfangszellen setzt der VDM dieses Bit, wenn geänderte Prozeßwerte empfangen werden. Die Bibliothek setzt das Bit

zurück, wenn alle geänderten Prozeßwerte der Datenzelle gelesen wurden.

- CZF_VALID
  Dieses Bit wird gesetzt, wenn mindestens ein Prozeßwert der Datenzelle einen gültigen Wert hat. Bleibt im Endgerät das Programm sehen (Anwenderprogramm läuft nicht), werden alle Empfangszellen ungültig.

SIEHE AUCH

VDM_Check_PV

7 API Referenz

Dieses Kapitel beschreibt die in der PAD-Interface Bibliothek definierten API Funktionen (Application Programming Interface). Das API unterteilt die Funktionen in drei logische Gruppen:

- PAD (Modul PADITF)

  Untere Kommunikationsschicht zum Datenaustausch;

- PDnet (Modul PADLIB)

  Zugriff auf logische PDnet-Funktionen (ab Session-Layer im Vergleich zum OSI-Referenzmodell);

- VDM (Modul PADVDM)

  Zugriff auf den VDM (Application-Layer im Vergleich zum OSI-Referenzmodell).

Die Bibliothek enthält folgende Funktionen.

Funktion	Beschreibung
Allgemeine Funktionen	
PAD_Result	Gibt den Status der letzten Operation zurück
PAD_Delay	Wartet eine bestimmte Zeit
Funktionen zum An- und Abmelden der lokalen PADs	
PAD_Init	Öffnet die Verbindung zu einem PAD
PAD_Select	Selektiert einen anderen PAD im lokalen Endgerät
PAD_Done	Schließt die Verbindung zum PAD
Funktionen zum Verwalten des lokalen PADs	
PAD_LifeCheck	Hält das PAD-Interface am Leben
PAD_CheckRestart	Prüft, ob der PAD einen Restart ausgelöst hat
PAD_GetResources	Liefert die Ressourcen vom aktuellen PAD
PAD_VersionInfo	Liefert Struktur auf die Firmwareversion im PAD
PAD_GetStatusLeds	Liefert Zustand der PAD Status Leuchtdioden
PAD_Read	Liest Daten aus dem PAD-Speicher
PAD_Write	Schreibt Daten in den PAD-Speicher
Funktionen zum Stellen und Lesen der PDnet Uhrzeit	
PAD_SetDateTime	Setzt Datum und Uhrzeit im PAD
PAD_DateTimeUpdated	Meldet Änderungen des Datums und der Zeit
PAD_GetDateTime	Liefert das Datum und die Uhrzeit des PAD
Funktionen zum Senden und Empfangen von Telegrammen	

Funktion	Beschreibung
PAD_SendTelegram	Sendet ein Telegramm
PAD_TxBufferEmpty	Prüft ob der Sendepuffer frei ist
PAD_ReceiveTelegram	Empfängt ein Telegramm
PAD_RxBufferEmpty	Prüft ob der Empfangspuffer frei ist
Funktionen der PDnet Lifeliste	
PAD_GetStationID	Liefert die Stationsadresse des lokalen PAD
PAD_GetGroupID	Liefert die Gruppenadresse des lokalen PAD
PDnet_LifeListChanged	Meldet Änderungen der PDnet Lifeliste
PDnet_OnLineNodeCout	Ermittelt die Anzahl der PADs die zur Zeit online sind
PDnet_OnLineStatus	Ermittelt, ob ein PAD im PDnet online ist
PDnet_ExtOnLineStatus	Ermittelt, über welche LAN-Kanäle ein PAD im PDnet erreichbar ist
PDnet_FlagByte	Liefert für einen PAD im PDnet das Flagbyte der Life-liste
PDnet_GroupMemebers	Liefert eine Liste aller Online Stationen die zu einer bestimmten Gruppe gehören
PDnet_GroupAdr	Liefert die Gruppenadresse eines PAD im PDnet
PDnet_ExtLifeList	Liefert die erweitere Lifeliste eines PDnet-Controllers
Funktionen zur Bearbeitung von VDM-Datenzellen	
VDM_Init_PV	Initialisiert die internen Strukturen für die VDM Funk-tionen
VDM_Done_PV	Gibt den von VDM_Init_PV angelegten Speicher frei
VDM_Read_PV	Liefert die nächste geänderte PV aus den VDM-Empfangszellen
VDM_Write_PV	Schreibt eine PV in alle zugehörigen VDM-Sendezellen
VDM_Check_PV	Überprüft, ob eine PV in einer VDM-Datenzelle exi-stiert
VDM_GetNextVdmHeader	Liefert den Kopf der nächsten VDM-Datenzelle

7.1 PAD_CheckRestart

AUFGABE

Prüft, ob der PAD einen Restart ausgelöst hat.

PASCAL DEFINITION

```
function PAD_CheckRestart : INT16;
```

C DEFINITION

```
INT16 PAD_CheckRestart(void);
```

RÜCKGABEWERT

Hat der lokale PAD keinen Restart ausgelöst, wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert. Z.B. CPDNET_RESTART_RUNNING, wenn der PAD-Restart läuft oder CPDNET_RESTART_END, wenn der PAD-Restart beendet ist.

BESCHREIBUNG

Diese Funktion überwacht, ob der lokale PAD einen Restart durchführt. Während des Restarts können keine Zuriffe auf den PAD statt finden. Die Funktion PAD_CheckRestart wird auch von der Funktion PAD_LifeCheck aufgerufen. Da alle PAD_???, PDnet_??? und VDM_??? Funktionen PAD_LifeCheck aufrufen, sollte nach jedem Funktionsaufruf der Rückgabewert oder die Funktion PAD_Result auf den Fehlercode CPDNET_RESTART_RUNNING oder CPDNET_RESTART_END überprüft werden.

Liefert eine Funktion CPDNET_RESTART_RUNNING sollte PAD_CheckRestart zyklisch aufgerufen werden, bis die Funktion CPDNET_RESTART_END meldet. Erst danach ist ein Zugriff auf den PAD wieder möglich.

HINWEIS

Während ein PAD einen Restart durchführt kann mit PAD_Select auf einen anderen vorhandenen lokalen PAD zugegriffen werden, wenn dieser keinen Restart durchführt.

C BEISPIEL

```
if (PAD_???() != CPDNET_OK)
{
  switch (PAD_Result())
  {
    case CPDNET_RESTART_RUNNING:
    {
      printf("PAD-Restart laeuft!\n");
      break;
    }
    case CPDNET_RESTART_END:
    {
      printf("PAD den Restart beendet!\n");
      break;
    }
    case CPDNET_ERR_RESTART:
    {
      printf("PAD hat einen FEHLERHAFTEN Restart"
             " duchgefuehrt!\n");
      exit(1);
    }
  }
}
```

SIEHE AUCH

PAD_LifeCheck

7.2 PAD_DateTimeUpdated

AUFGABE

Meldet Änderungen des Datums und der Zeit.

PASCAL DEFINITION

```
function PAD_DateTimeUpdated(
  var Flag: INT16
) : INT16;
```

C DEFINITION

```
INT16 PAD_DateTimeUpdated(
  INT16 *Flag
);
```

AUSGABEWERTE

- Flag
  Die zurückgegebene Variable Flag enthält eine 1, wenn die Uhrzeit im lokalen PAD verändert wurde. Andernfalls enthält Flag eine 0.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Diese Funktion meldet, ob sich die Uhrzeit bzw. das Datum im PAD geändert hat. Bei Änderungen der PAD Uhrzeit sollte das Anwenderprogramm die Systemzeit im Endgerät stellen.

7.3 PAD_Delay

AUFGABE

PAD_Delay wartet eine bestimmte Zeit.

PASCAL DEFINITION

```
procedure PAD_Delay(ms: UINT16);
```

C DEFINITION

```
void PAD_Delay(UINT16 ms);
```

EINGABEWERTE

- In ms wird die Zeit in ms übergeben, die gewartet werden soll.

RÜCKGABEWERT

- keiner

BESCHREIBUNG

Die Funktion wartet bis mindestens die übergebene Zeit verstrichen ist und kehrt dann zurück. Je nach Betriebssystem wird eine Sleep Funktion aufgerufen oder eine Warteschleife durchlaufen.

HINWEIS

Da die Auflösung der Zeitfunktionen bei einigen Betriebssystemen größer als 1 ms ist, kann die Verzögerung der Funktion PAD_Delay länger dauern, als die übergebene Zeit.

7.4 PAD_Done

AUFGABE

PAD_Done schließt die Verbindung zum PAD.

PASCAL DEFINITION

```
function PAD_Done : INT16;
```

C DEFINITION

```
INT16 PAD_Done(void);
```

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Beim Beenden der Anwendung sollten alle PAD Verbindungen mit PAD_Done geschlossen werden. Nach dem Aufruf dieser Funktion darf keine andere API-Funktion mehr aufgerufen werden.

SIEHE AUCH

PAD_Init

7.5 PAD_GetDateTime

AUFGABE

Liest Datum und Uhrzeit aus dem lokalen PAD.

PASCAL DEFINITION

```
function PAD_GetDateTime(
  var PadDate: tPadDate;
  var PadTime: tPadTime
) : INT16;
```

C DEFINITION

```
INT16 PAD_GetDateTime(
  tPadDate _*PadDate,
  tPadTime _*PadTime
);
```

AUSGABEWERTE

- PadDate
  Über die Struktur PadDate wird das Datum des lokalen PAD zurückgegeben. Der Speicher für die Struktur PadDate muß vor dem Aufruf der Funktion angelegt werden.
- PadTime
  Über die Struktur PadTime wird die Uhrzeit des lokalen PAD zurückgegeben. Der Speicher für die Struktur PadTime muß vor dem Aufruf der Funktion angelegt werden.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Wenn im PAD eine gültige Uhrzeit steht, liefert die Funktion das Datum und die Zeit aus dem PAD

SIEHE AUCH

tPadDate, tPadTime, PAD_SetDateTime, PAD_DateTimeUpdated

7.6 PAD_GetGroupID

AUFGABE

Liefert die Gruppenadresse des lokalen PAD.

PASCAL DEFINITION

```
function PAD_GetGroupID(
  var GroupID: UINT8
) : INT16;
```

C DEFINITION

```
INT16 PAD_GetGroupID(
  UINT8 GroupID
);
```

AUSGABEWERTE

- GroupID
  Über GroupID wird die Gruppenadresse des lokalen PAD zurückgeliefert.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Die Funktion PAD_GetGroupID liest die aktuelle Gruppenadresse aus dem lokalen PAD. Nach einem Restart kann die Gruppenadresse verändert sein, wenn z.B. im NetPro die Station auf eine andere Gruppenadresse gelegt wurden.

SIEHE AUCH

PAD_GetStationID, PDnet_GroupAdr

7.7 PAD_GetResources

AUFGABE

Liefert die verwendeten Ressourcen vom aktuellen PAD.

DEFINITION

Da jede Plattform andere Ressourcen verwendet, stehen die Definitionen von PAD_GetResources in den folgenden Kapiteln.

BESCHREIBUNG

Die Funktion PAD_GetResources ermittelt verwendeten Ressourcen vom aktuellen PAD aus den Parametern von PAD_Init oder den eingestellten Ressourcen des Gerätetreibers. Durch die Funktion kann ein Anwenderprogramm die eingestellten Ressourcen anzeigen, um diese mit den Hardwareeinstellungen auf dem PAD zu vergleichen. Außerdem kann ermittelt werden, welche Ressourcen der aktuelle PAD verwendet, wenn sich mehrere PADs im lokalen Endgerät befinden.

SIEHE AUCH

PAD_Init, PAD_Select

7.7.1 IBM-PC DOS (Real Mode)

PASCAL DEFINITION

```
function PAD_GetResources(
  var irqno:   UINT8;
  var iobase:  UINT16;
  var membase: UINT32;
  var memlen:  UINT32
) : INT16;
```

C DEFINITION

```
INT16 PAD_GetResources(
  UINT8  *irqno,
  UINT16 *iobase,
  UINT32 *membase,
  UINT32 *memlen
);
```

AUSGABEWERTE

- irqno
  Über irqno wird der eingestellte IRQ zurückgegeben.
- iobase
  Über iobase wird die eingestellte I/O-Port Adresse zurückgegeben.
- membase
  Über membase wird die eingestellte Basisadresse des Speicherfensters zurückgegeben.

- memlen
  Über memlen wird die ermittelte Größe des Speicherfensters zurückgegeben.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

7.7.2 IBM-PC DOS (Protected Mode)

DEFINITION

Siehe PAD_Init für DOS (Real Mode)

7.7.3 IBM-PC Windows 3.1

DEFINITION

Siehe PAD_Init für DOS (Real Mode)

7.7.4 IBM-PC Windows 95

DEFINITION

Siehe PAD_Init für DOS (Real Mode)

7.7.5 IBM-PC Windows NT 3.5x/4.00

DEFINITION

Siehe PAD_Init für DOS (Real Mode)

7.7.6 IBM-PC OS/2 Warp 3/4

DEFINITION

Siehe PAD_Init für DOS (Real Mode)

7.7.7 IBM-PC QNX 3.21 (Protected Mode)

DEFINITION

Siehe PAD_Init für DOS (Real Mode)

7.7.8 IBM-PC QNX 4.22 (Protected Mode)

DEFINITION

Siehe PAD_Init für DOS (Real Mode)

7.7.9 DEC AlphaStation OpenVMS/AXP 6.2/7.0

DEFINITION

Siehe PAD_Init für DOS (Real Mode)

7.7.10 SUN IPC/IPX (SUN-OS 4.1.2)

C DEFINITION

```
INT16 PAD_GetResources(
  UINT8 *slotnr
);
```

AUSGABEWERTE

- slotnr
  Über slotnr wird die Steckplatznummer des SBus zurückgegeben, auf dem der aktuelle PAD-SBus steckt.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

7.7.11 Motorola-8420 (System V/m88k)

Funktion nicht vorhanden.

7.8 PAD_GetStationID

AUFGABE

Liefert die Stationsadresse des lokalen PAD.

PASCAL DEFINITION

```
function PAD_GetStationID(
  var StationID: UINT8
) : INT16;
```

C DEFINITION

```
INT16 PAD_GetStationID(
  UINT8 *StationID
);
```

AUSGABEWERTE

- StationID
  Über StationID wird die Stationsadresse des lokalen PAD zurückgeliefert.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Die Funktion PAD_GetStationID liest die aktuelle Stationsadresse aus dem lokalen PAD. Nach einem Restart kann die Stationsadresse verändert sein, wenn z.B. im NetPro die Station auf eine andere Stationsadresse gelegt wurden.

SIEHE AUCH

PDnet_OnLineStatus, PDnet_ExtOnLineStatus, PDnet_ExtLifeList

7.9 PAD_GetStatusLeds

AUFGABE

Liefert Zustand der PAD Status Leuchtdioden.

PASCAL DEFINITION

```
function PAD_GetStatusLeds(
  var PadStatusLeds: UINT16;
  var PadErrorsLeds: UINT16
) : INT16;
```

C DEFINITION

```
INT16 PAD_GetStatusLeds(
  UINT16 *PadStatusLeds,
  UINT16 *PadErrorsLeds
);
```

AUSGABEWERTE

- PadStatusLeds
  Über PadStatusLeds werden die CLED_??? Statusbits geliefert. Die jeweiligen Statusbits sind 0, wenn die Option deaktiv ist und 1, wenn die Option aktiv ist.
- PadErrorLeds
  Über PadErrorLeds werden die CLED_??? Fehlerbits geliefert. Die jeweiligen Fehlerbits sind 0, wenn die Option OK ist und 1, wenn die Option einen Fehler entdeckt hat.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Die Funktion liefert den PAD Status, der bei PDnet-Controllern für die SPS in den gelben Leuchtdioden angezeigt wird. Ist ein Bit in PadErrorsLeds gesetzt, blinkt die entsprechende Leuchtdiode. Andernfalls legt PadStatusLeds fest, das die Leuchtdiode bei gesetztem Bit dauernd leuchtet oder bei rückgesetztem Bit nicht leuchtet.

SIEHE AUCH

CLED_???

7.10 PAD_LifeCheck

AUFGABE

Hält das PAD-Interface am Leben.

PASCAL DEFINITION

```
function PAD_LifeCheck: INT16;
```

C DEFINITION

```
INT16 PAD_LifeCheck(void);
```

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Damit der PAD prüfen kann, ob das Anwenderprogramm im Endgerät läuft, überwacht der PAD das Interface zum System. Zu diesem Zweck existieren einige Variablen in der Statusseite des PAD, welche zyklisch von der Applikation nachgetriggert werden müssen (Software Watchdog). Diese Funktionen werden komplett von der Funktion PAD_LifeCheck übernommen. Alle PAD_???, PDnet_??? und VDM_??? Funktionen rufen PAD_LifeCheck auf.

ACHTUNG

Diese Funktion muß im Programm zyklisch aufgerufen werden! Der maximale Zeitabstand zwischen den Aufrufen von PAD_LifeCheck wird beim Aufruf von PAD_Init im Parameter PcCheckTmtVal angegeben. Es empfiehlt sich, die Routine in der Hauptwarteschleife des Programmes (z. B.: repeat until keypressed) aufzurufen oder ein Timerprozeß zu starten, der PAD_LifeCheck periodisch aufruft.

SIEHE AUCH

PAD_Init, PAD_Done, PAD_CheckRestart

7.11 PAD_Init

AUFGABE

Die Funktion PAD_Init öffnet die Verbindung zum lokalen PAD.

DEFINITION

Da jede Plattform andere Ressourcen verwendet, stehen die Definitionen von PAD_Init in den folgenden Kapiteln.

BESCHREIBUNG

Die Funktion öffnet die Verbindung zwischen der Applikation und dem PAD. Dazu prüft die Bibliothek, ob an übergebenen Ressourcen im PAD-Speicher die Strings „APEX“ und „PAD“ gefunden werden. Wurde der PAD gefunden, gibt PAD_Init CPDNET_OK zurück. Anderenfalls wird ein CPDENT_??? Fehlercode zurückgegeben.

ACHTUNG

PAD_Init muß vor dem Aufruf einer anderen API-Funktion aufgerufen werden! Danach können die API-Funktionen auf den lokalen PAD zugreifen, bis die Verbindung mit PAD_Done getrennt wird.

SIEHE AUCH

PAD_Done, PAD_Select

7.11.1 IBM-PC DOS (Real Mode)

PASCAL DEFINITION

```
function PAD_Init(
  aIrqNo:        UINT8;
  aMemSeg:       UINT16;
  aIOBase:       UINT16;
  PcCheckTmrVal: UINT16;
  var handle:    UINT16
) : UINT16;
```

C DEFINITION

```
INT16 PAD_Init(
  UINT8   aIrqNo,
  UINT16  aMemSeg,
  UINT16  aIOBase,
  UINT16  PcCheckTmrVal,
  UINT16 *handle
);
```

EINGABEWERTE

- aIrqNo
  In aIrqNo wird die auf dem PAD eingestellte IRQ Adresse übergeben. Der Parameter wird zur Zeit noch nicht von der Bibliothek ausgewertet.

- aMemSeg
  In aMemSeg wird das Speichersegment des lokalen PAD übergeben. Die Adresse muß mit den Einstellungen der DIP-Schalter SW1 auf dem PAD übereinstimmen.
- aIOBase
  In aIOBase wird die I/O-Port Adresse des lokalen PAD übergeben. Die Adresse muß mit den Einstellungen der DIP-Schalter SW1 auf dem PAD übereinstimmen.
- PcCheckTmrVal
  PcCheckTmrVal setzt die Zeit in 10ms-Schritten, innerhalb der PAD die laufende Anwendung überwacht. Die Anwendung muß die Funktion PAD_LifeCheck innerhalb dieser Zeit aufrufen, damit der PAD ein laufendes Anwenderprogramm erkennt.

AUSGABEWERTE

- handle
  Über handle wird eine eindeutiger Wert zurückgegeben, mit dem die Funktion PAD_Select den initialisierten PAD wählen kann. Der handle wird benötigt, wenn die PAD-Interface Bibliothek mehrere PADs im lokalen Endgerät verwalten soll. Befindet sich nur ein PAD im lokalen Endgerät, kann handle vernachlässigt werden.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

PASCAL BEISPIEL

```
program Demo;

  procedure PAD_Initialisieren;
  var
    handle: UINT16;
  begin
    if (PAD_Init(10, $D000, $290, 1000, handle)
    <> CPDNET_OK then
    begin
      writeln('Status der PAD-Operation: ',
              PAD_Result);
      halt(1);
    end;
  end;

begin
  PAD_Initialisieren;
  ...
  PAD_Done;
end.
```


C BEISPIEL

```
int main(int argc, char **argv)
{
  UINT16 handle;

  /* PAD-PC initialisieren */
  if (PAD_Init(10, 0xD000, 0x290, 1000, &handle)
    != CPDNET_OK)
  {
    printf("Status der PAD-Operation: %d\n",
           PAD_Result());
    exit(1);
  }

  /* PAD-Funktionen aufrufen */
  ...

  /* PAD-Verbindung schließen */
  PAD_Done();
  return 0;
}
```

7.11.2 IBM-PC DOS (Protected Mode)

DEFINITION

Siehe PAD_Init für DOS (Real Mode)

7.11.3 IBM-PC Windows 3.1

DEFINITION

Siehe PAD_Init für DOS (Real Mode)

7.11.4 IBM-PC Windows 95

PASCAL DEFINITION

```
function PAD_Init(
  padno:         UINT8;
  PcCheckTmrVal: UINT16,
  var handle:    UINT16
) : UINT16;
```

C DEFINITION

```
INT16 PAD_Init(
  const UINT8 padno,
  UINT16      PcCheckTmrVal,
  UINT16     *handle
);
```

EINGABEWERTE

- padno
  Über padno wird die Gerätenummer des lokalen PADs übergeben, auf den zugegriffen werden soll. Die Gerätenummer kann zwischen 1 und CPAD_COUNT liegen. Bei padno=0 verwendet die Funktion den PDnet-Controller mit der kleinsten vorhandenen Gerätenummer.
- PcCheckTmrVal
  PcCheckTmrVal setzt die Zeit in 10ms-Schritten, innerhalb der

PAD die laufende Anwendung überwacht. Die Anwendung muß die Funktion PAD_LifeCheck innerhalb dieser Zeit aufrufen, damit der PAD ein laufendes Anwenderprogramm erkennt.

AUSGABEWERTE

- handle
  Über handle wird ein eindeutiger Wert zurückgegeben, mit dem die Funktion PAD_Select den initialisierten PAD wählen kann. Der handle wird benötigt, wenn die PAD-Interface Bibliothek mehrere PADs im lokalen Endgerät verwalten soll. Befindet sich nur ein PAD im lokalen Endgerät, kann handle vernachlässigt werden.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

HINWEIS

Die Gerätenummer wird im Windows 95 Geräte-Manager vor jedem “APEX PDnet-Controller“ in eckigen Klammern angezeigt.

PAD_Init kann mehrmals mit verschiedenen Gerätenummern aufgerufen werden, wenn mehrere PDnet-Controller im lokalen PC installiert sind.

Die vom PAD verwendeten Ressourcen werden vom Gerätetreiber aus der Registrierdatenbank ermittelt.

7.11.5 IBM-PC Windows NT 3.5x/4.00

DEFINITION

Siehe PAD_Init für Windows 95

HINWEIS

Die Gerätenummer wird in den Windows NT Netzwerkeinstellungen bei den Netzwerkkarten vor jedem “APEX PDnet-Controller“ in eckigen Klammern angezeigt.

7.11.6 IBM-PC OS/2 Warp 3/4

DEFINITION

Siehe PAD_Init für Windows 95

HINWEIS

Zur Zeit wird nur ein lokaler PDnet-Controller mit der Gerätenummer 1 unterstützt.

Die vom PAD verwendeten Ressourcen werden vom Gerätetreiber aus der Datei CONFIG.SYS ermittelt.

7.11.7 IBM-PC QNX 3.21 (Protected Mode)

DEFINITION

Siehe PAD_Init für DOS (Real Mode)

7.11.8 IBM-PC QNX 4.22 (Protected Mode)

DEFINITION

Siehe PAD_Init für DOS (Real Mode)

7.11.9 DEC AlphaStation OpenVMS/AXP 6.2/7.0

C DEFINITION

```
INT16 PAD_Init(
  char   *devicename,
  UINT16  PcCheckTmrVal,
  UINT16 *handle
);
```

EINGABEWERTE

- devicename
  In devicename wird der Name des OpenVMS Device für den geladenen Gerätetreiber übergeben.
- PcCheckTmrVal
  PcCheckTmrVal setzt die Zeit in 10ms-Schritten, innerhalb der PAD die laufende Anwendung überwacht. Die Anwendung muß die Funktion PAD_LifeCheck innerhalb dieser Zeit aufrufen, damit der PAD ein laufendes Anwenderprogramm erkennt.

AUSGABEWERTE

- handle
  Über handle wird eine eindeutiger Wert zurückgegeben, mit dem die Funktion PAD_Select den initialisierten PAD wählen kann. Der handle wird benötigt, wenn die PAD-Interface Bibliothek mehrere PADs im lokalen Endgerät verwalten soll. Befindet sich nur ein PAD im lokalen Endgerät, kann handle vernachlässigt werden.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

C BEISPIEL

```
int main(int argc, char **argv)
{
  UINT16 handle;

  /* PAD-PC initialisieren auf Gerät JDA0: */
  if (PAD_Init(“JDA0:“, 1000, &handle)
    != CPDNET_OK)
  {
    printf("Status der PAD-Operation: %d\n",
           PAD_Result());
    exit(1);
  }

  /* PAD-Funktionen aufrufen */
  ...

  /* PAD-Verbindung schließen */
  PAD_Done();
  return 0;
}
```

7.11.10 SUN IPC/IPX (SUN-OS 4.1.2)

C DEFINITION

```
INT16 PAD_Init(
  UINT8   aSlotNum,
  UINT16  PcCheckTmrVal,
  UINT16 *handle
);
```

EINGABEWERTE

- aSlotNum
  In aSlotNum wird die Steckplatznummer des SBus übergeben, auf dem sich der PAD-SBus befindet.
- PcCheckTmrVal
  PcCheckTmrVal setzt die Zeit in 10ms-Schritten, innerhalb der PAD die laufende Anwendung überwacht. Die Anwendung muß die Funktion PAD_LifeCheck innerhalb dieser Zeit aufrufen, damit der PAD ein laufendes Anwenderprogramm erkennt.

AUSGABEWERTE

- handle
  Über handle wird eine eindeutiger Wert zurückgegeben, mit dem die Funktion PAD_Select den initialisierten PAD wählen kann. Der handle wird benötigt, wenn die PAD-Interface Bibliothek mehrere PADs im lokalen Endgerät verwalten soll. Befindet sich nur ein PAD im lokalen Endgerät, kann handle vernachlässigt werden.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

C BEISPIEL

```
int main(int argc, char **argv)
{
  UINT16 handle;

  /* PAD-SBus initialisieren auf SBus-Slot 2 */
  if (PAD_Init(2, 1000, &handle) != CPDNET_OK)
  {
    printf("Status der PAD-Operation: %d\n",
           PAD_Result());
    exit(1);
  }

  /* PAD-Funktionen aufrufen */
  ...

  /* PAD-Verbindung schließen */
  PAD_Done();
  return 0;
}
```

7.11.11 SUN Ultra 5 (Solaris 2.6)

C DEFINITION

```
INT16 PAD_Init(
  UINT8   aSlotNum,
  UINT16  PcCheckTmrVal,
  UINT16 *handle
);
```

EINGABEWERTE

- aSlotNum
  Jeder installierte PAD-PCI meldet sich unter einen Gerätenamen „`/dev/pdnetX`“ beim Betreibssystem an, wenn der Gerätetreiber pdnet geladen wird. In aSlotNum wird die Gerätenummer X im Gerätenamen angegeben.
- PcCheckTmrVal
  PcCheckTmrVal setzt die Zeit in 10ms-Schritten, innerhalb der PAD die laufende Anwendung überwacht. Die Anwendung muß die Funktion PAD_LifeCheck innerhalb dieser Zeit aufrufen, damit der PAD ein laufendes Anwenderprogramm erkennt.

AUSGABEWERTE

- handle
  Über handle wird eine eindeutiger Wert zurückgegeben, mit dem die Funktion PAD_Select den initialisierten PAD wählen kann. Der handle wird benötigt, wenn die PAD-Interface Bibliothek mehrere PADs im lokalen Endgerät verwalten soll. Befindet sich nur ein PAD im lokalen Endgerät, kann handle vernachlässigt werden.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

C BEISPIEL

```
int main(int argc, char **argv)
{
  UINT16 handle;

  /* PAD-PCI auf Gerät “/dev/pdnet2“ */
  if (PAD_Init(2, 1000, &handle) != CPDNET_OK)
  {
    printf("Status der PAD-Operation: %d\n",
           PAD_Result());
    exit(1);
  }

  /* PAD-Funktionen aufrufen */
  ...

  /* PAD-Verbindung schließen */
  PAD_Done();
  return 0;
}
```

7.11.12 Motorola-8420 (System V/m88k)

C DEFINITION

```
INT16 PAD_Init(
  UINT8   aBlockNr,
  UINT16  PcCheckTmrVal,
  UINT16 *handle
);
```

EINGABEWERTE

- In aBlockNr wird der Speicherblock in 256kByte-Schritten übergeben, auf dem sich der PAD-VME befindet.
- PcCheckTmrVal setzt die Zeit in 10ms-Schritten, innerhalb der PAD die laufende Anwendung überwacht. Die Anwendung muß die Funktion PAD_LifeCheck innerhalb dieser Zeit aufrufen, damit der PAD ein laufendes Anwenderprogramm erkennt.

AUSGABEWERTE

- handle
  Über handle wird eine eindeutiger Wert zurückgegeben, mit dem die Funktion PAD_Select den initialisierten PAD wählen kann. Der handle wird benötigt, wenn die PAD-Interface Bibliothek mehrere PADs im lokalen Endgerät verwalten soll. Befindet sich nur ein PAD im lokalen Endgerät, kann handle vernachlässigt werden.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

C BEISPIEL

```
int main(int argc, char **argv)
{
  UINT16 handle;

  /* VME-PAD initialisieren auf Adr.0x02000000 */
  /* 32MByte = 128 * 256kByte */
  if (PAD_Init(128, 100, &handle) != CPDNET_OK)
  {
    /* Fehler aufgetreten */
    printf("Status der PAD-Operation: %d\n",
           PAD_Result());
    exit(1);
  }

  /* PAD-Funktionen aufrufen */
  ...

  /* PAD-Verbindung schließen */
  PAD_Done();
  return 0;
}
```


7.12 PAD_Read

AUFGABE

Kopiert Bytes vom PAD-Speicher zum Systemspeicher.

PASCAL DEFINITION

```
function PAD_Read(
  dest: pUINT8;
  src:  UINT32;
  n:    UINT16
) : UINT16;
```

C DEFINITION

```
UINT16 PAD_Read(
  UINT8        *dest,
  const UINT32  src,
  UINT16        n
);
```

EINGABEWERTE

- src
  In src wird die Adresse des PAD-Speichers angegeben, ab der die Daten gelesen werden.
- n
  In n steht die Anzahl der zu kopierenden Bytes.

AUSGABEWERTE

- dest
  In dest wird die Systemadresse übergeben, wohin die Daten kopiert werden.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Die Funktion ermöglicht das Auslesen des PAD-Speichers über die PAD-Adressen.

C BEISPIEL

```
/* 100 Bytes lesen */
UINT16 count = 100;
UINT8  *m;
/* Dynamischen Speicher anfordern */
if ((m = (UINT8 *) malloc(count)) == NULL)
{
  printf("Zu wenig Speicher frei!\n");
  return;
}
/* Daten von PAD-Adresse 0x0000 lesen */
if (PAD_Read(m, 0x0000, count) != CPDNET_OK)
{
  printf("Fehler bei PAD_Read()!\n");
  free(m);
  return;
```



```
}
/* Speicherbereich anzeigen */
...
free(m);
```

SIEHE AUCH

PAD_Write, PAD_Addr

7.13 PAD_ReceiveTelegram

AUFGABE

Empfängt ein Telegramm von einem PDnet-Controller.

PASCAL DEFINITION

```
function PAD_ReceiveTelegram(
  var telegram: tTelegram
) : INT16;
```

C DEFINITION

```
INT16 PAD_ReceiveTelegram(
      tTelegram   *telegram
);
```

AUSGABEWERTE

- telegram
  In telegram wird die Adresse des zu empfangenen Telegramms übergeben. Die Felder des Telegramms entnehmen Sie dem Typ tTelegramm.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert. Z.B. CPDNET_RX_EMPTY, wenn kein Telegramm im Empfangspuffer verfügbar ist.

BESCHREIBUNG

Diese Funktion übernimmt ein im PAD bereitstehendes Empfangstelegramm und kopiert es nach telegramm.

ACHTUNG

Das Anwenderprogramm muß zyklisch alle empfangenen Telegramme über PAD_ReceiveTelegramm abholen, damit es zu keinen „Verstopfungen" im Netz kommt.

PASCAL BEISPIEL

```
procedure EmpfangeTelegramme;
var
  tg: tTelegram;
begin
  tg.Semaphore := 0;
  tg.Len       := 0;
  if PAD_ReceiveTelegram(tg) = CPDNET_OK then
  begin
    case Tg.Typ of
```

```
    ...
    end;
  end;
end;
```

C BEISPIEL

```
void EmpfangeTelegramme(void)
{
  static tTelegram tg = 0;
  tg.Semaphore = 0;
  tg.Len       = 0;
  if (PAD_ReceiveTelegram(&tg)==CPDNET_OK)
  {
    switch (tg.Typ)
    {
      ...
    }
  }
}
```

SIEHE AUCH

PAD_RxBufferEmpty, PAD_SendTelegram

7.14 PAD_Result

AUFGABE

Gibt den Status der letzten Operation zurück und löscht den Status.

PASCAL DEFINITION

```
function PAD_Result : INT16;
```

C DEFINITION

```
INT16 PAD_Result(void);
```

RÜCKGABEWERT

- Ist in der Bibliothek kein Fehler aufgetreten, wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Wenn bei einer Funktion Fehler auftritt, so wird der Fehlercode in einer internen Variable gespeichert und von der Funktion zurückgegeben. Mit der Funktion PAD_Result kann der letzte Fehlercode gelesen und auf 0 zurückgesetzt werden.

C BEISPIEL

```
UINT8 Result = PAD_Result();
printf("Status der letzten PAD-Operation: %d ",
       Result);
switch (Result)
{
  case CPDNET_OK:
  {
    printf("= CPDNET_OK");
    break;
  }
  case CPDNET_NO_INIT:
  {
    printf("= CPDNET_NO_INIT");
    break;
  }
  case CPDNET_NO_CONTROLLER_DETECTED:
  {
    printf("= CPDNET_NO_CONTROLLER_DETECTED");
    break;
  }
  case CPDNET_BAD_PARAM:
  {
    printf("= CPDNET_BAD_PARAM");
    break;
  }
}
printf("\n");
```

7.15 PAD_RxBufferEmpty

AUFGABE

Prüft, ob ein Telegramm empfangen wurde.

PASCAL DEFINITION

```
function PAD_RxBufferEmpty: INT16;
```

C DEFINITION

```
INT16 PAD_RxBufferEmpty(void);
```

RÜCKGABEWERT

- Bei einem leeren Empfangspuffer wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert. Z.B. CPDNET_RX_FULL, wenn ein neues Telegramm im Empfangspuffer liegt.

BESCHREIBUNG

Die Funktion meldet, ob ein neues Telegramm im Empfangspuffer angekommen ist. Wenn dies der Fall ist, sollte das Telegramm mit der Funktion PAD_ReceiveTelegram gelesen werden, damit es zu keinen „Verstopfungen" im Netz kommt.

HINWEIS

Diese Funktion ist wegen der Kompatibilität zur alten Bibliothek noch vorhanden. Die Funktion PAD_RxBufferEmpty braucht aber nicht verwendet werden, da die Funktion PAD_ReceiveTelegram einen Fehler meldet, wenn der Empfangspuffer leer ist.

PASCAL BEISPIEL

```
if PAD_RxBufferEmpty=CPDNET_RX_FULL then
begin
  PAD_ReceiveTelegram(tg);
  case tg.Typ of
    ...
  end;
end;
```

C BEISPIEL

```
if (PAD_RxBufferEmpty()==CPDNET_RX_FULL)
{
  PAD_ReceiveTelegram(&tg);
  switch (tg.Typ)
  {
    ...
  }
}
```

SIEHE AUCH

PAD_ReceiveTelegram

7.16 PAD_Select

AUFGABE

Selektiert einen anderen PAD im lokalen Endgerät.

PASCAL DEFINITION

```
function PAD_Select(
  handle: UINT16
) : INT16;
```

C DEFINITION

```
INT16 PAD_Select(
  UINT16 handle
);
```

EINGABEWERTE

- In handle wird der von PAD_Init gelieferte handle für den zu selektierenden PAD übergeben.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Die Bibliothek unterstützt für einige Plattformen mehrere PADs im lokalen Endgerät. Mit der Funktion PAD_Select wird der PAD gewählt, auf den die Funktionen der Bibliothek zugreifen.

ACHTUNG

Jeder zu wählende PAD muß vorher einmalig mit der Funktion PAD_Init initialisiert werden.

SIEHE AUCH

PAD_Init

7.17 PAD_SendTelegram

AUFGABE

Sendet ein Telegramm an einen PDnet-Controller.

PASCAL DEFINITION

```
function PAD_SendTelegram(
  var telegram: tTelegram
) : INT16;
```

C DEFINITION

```
INT16 PAD_SendTelegram(
      tTelegram   *telegram
);
```

EINGABEWERTE

- telegram
  In telegram wird die Adresse des zu sendenden Telegramms übergeben. Die Felder des Telegramms entnehmen Sie dem Typ tTelegramm.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert. Z.B. CPDNET_TX_FULL, wenn für das Telegramm kein freier Sendepuffer verfügbar ist.

BESCHREIBUNG

Die Funktion sendet ein Telegramm zu einem anderen PDnet-Controller oder einer Gruppe von PDnet-Controllern.

ACHTUNG

Liefert die Funktion CPDNET_OK, so handelt es sich dabei um keine Sendebestätigung, sondern nur um eine Bestätigung, das das Telegramm in einen freien Sendepuffer im PAD kopiert wurde.

PASCAL BEISPIEL

```
{ Neustart eines Teilnehmers auslösen }
function PAD_Neustart(PDnetId: byte): boolean;
var
  tg: tTelegram;
begin
  PAD_Neustart := FALSE;
  if PDnetId = 0 then
    exit;
  fillchar(tg, sizeof(tg),#0);
  tg.PDnetId := PDnetId;
  tg.Typ     := 250;
  tg.Len     := 3;
  tg.Data[0] := 255;
  tg.Data[1] := 1;
  PAD_Neustart :=
   (PAD_SendTelegram(tg) = CPDNET_OK);
end;
```


C BEISPIEL

```
/* Neustart eines Teilnehmers auslösen */
int PAD_Neustart(UINT8 PDnetId)
{
  tTelegram tg;
  if (PDnetId==0)
    return 0;
  memset(&tg, 0, sizeof(tg));
  tg.PDnetId = PDnetId;
  tg.Typ     = 250;
  tg.Len     = 3;
  tg.Data[0] = 255;
  tg.Data[1] = 1;
  if (PAD_SendTelegram(&tg) == CPDNET_OK)
    return 1;
  else
    return 0;
}
```

SIEHE AUCH

PAD_TxBufferEmpty, PAD_ReceiveTelegram

7.18 PAD_SetDateTime

AUFGABE

Setzt das Datum und die Uhrzeit in einem PAD.

PASCAL DEFINITION

```
function PAD_SetDateTime(
  PDnetId: UINT8;
  var PadDate: tPadDate;
  var PadTime: tPadTime
) : INT16;
```

C DEFINITION

```
INT16 _PAD_SetDateTime(
  UINT8      PDnetId,
  tPadDate _*PadDate,
  tPadTime _*PadTime
);
```

EINGABEWERTE

- PDnetId
  PDnetId enthält die Stationsadresse, an die ein Uhrzeittelegramm gesendet werden soll. Ist PDnetId 0, wird das Uhrzeittelegramm an alle Stationen im PDnet gesendet.
- PadDate
  In der Struktur PadDate steht das neue Datum, das in den PAD geschrieben wird.
- PadTime
  In der Struktur PadTime steht die neue Zeit, die in den PAD geschrieben wird.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Die Funktion schreibt das übergebene Datum und die Zeit in den angegebenen PAD. Danach ist das Datum und die Zeit im PAD gültig. Über die Funktion erhalten alle PDnet-Controller im Netz eine einheitliche Zeit. Die Endgeräte können diese Zeit auslesen, um die Uhrzeit aller Stationen zu synchronisieren.

SIEHE AUCH

tPadDate, tPadTime, PAD_GetDateTime, PAD_DateTimeUpdated

7.19 PAD_TxBufferEmpty

AUFGABE

Prüft, ob der Sendepuffer frei ist.

PASCAL DEFINITION

```
function PAD_TxBufferEmpty: INT16;
```

C DEFINITION

```
INT16 PAD_TxBufferEmpty(void);
```

RÜCKGABEWERT

- Bei einem leeren Sendepuffer wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert. Z.B. CPDNET_TX_FULL, wenn der Sendepuffer noch belegt ist.

BESCHREIBUNG

Diese Funktion sollte vor dem Senden eines Telegramms aufgerufen werden, damit dieses nicht verworfen wird, wenn der Sendepuffer belegt ist.

HINWEIS

Diese Funktion ist wegen der Kompatibilität zur alten Bibliothek noch vorhanden. Die Funktion PAD_TxBufferEmpty braucht aber nicht verwendet werden, da die Funktion PAD_SendTelegram einen Fehler meldet, wenn der Sendepuffer noch belegt ist.

PASCAL BEISPIEL

```
if PAD_TxBufferEmpty = CPDNET_OK then
begin
  { Telegramm aufbauen }
  ...
  PAD_SendTelegram(tg);
end;
```

C BEISPIEL

```
if (PAD_TxBufferEmpty() == CPDNET_OK)
{
  /* Telegramm aufbauen */
  ...
  PAD_SendTelegram(&tg);
}
```

SIEHE AUCH

PAD_SendTelegram

7.20 PAD_VersionInfo

AUFGABE

Liefert eine Struktur mit der PAD Firmwareversion.

PASCAL DEFINITION

```
function PAD_VersionInfo(
  var vi: tPAD_VersionInfo
) : INT16;
```

C DEFINITION

```
INT16 PAD_VersionInfo(
  tPAD_VersionInfo *vi
);
```

AUSGABEWERTE

- vi
  Über vi wird die Struktur mit der Firmwareversion des lokalen PAD zurückgegeben. Der Speicher für die Struktur vi muß vor dem Aufruf der Funktion angelegt werden.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Die Firmwareversion wird über ein Firmwareinfo Telegramm angefordert und von der Funktion zurückgegeben.

C BEISPIEL

```
tPAD_VersionInfo vi;
if (PAD_VersionInfo(&vi) != CVDM_OK)
{
  printf("Fehler bei PAD_VersionInfo()!\n");
  return;
}
/* Firmwareversion ungültig? */
if ((vi->FwDay > 0x31) || (vi->FwMonth> 0x12))
{
  return;
}

printf("Firmwareversion: %02x.%02x vom"
       " %02x.%02x.19%02x\n",
        vi->FwVersion,
        vi->FwRelease,
        vi->FwDay,
        vi->FwMonth,
        vi->FwYear);
```

7.21 PAD_Write

AUFGABE

Kopiert Bytes vom Systemspeicher zum PAD-Speicher.

PASCAL DEFINITION

```
function PAD_Write(
  dest: UINT32;
  src:  pUINT8;
  n:    UINT16
) : UINT16;
```

C DEFINITION

```
UINT16 PAD_Write(
  UINT32       dest,
  const UINT8 *src,
  UINT16       n
);
```

EINGABEWERTE

- dest
  In dest wird die Adresse des PAD-Speichers angegeben, wohin die Daten kopiert werden.
- src
  In src wird die Systemadresse übergeben, ab der die Daten gelesen werden.
- n
  In n steht die Anzahl der zu kopierenden Bytes.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Diese Funktion ermöglicht das Beschreiben des PAD-Speichers über die PAD-Adressen.

ACHTUNG

Die Funktion wird nur intern in der Bibliothek verwendet. Der Anwender sollte PAD_Write nicht benutzen, da ein Überschreiben des PAD-Speichers an der falschen Stelle die Firmware des PAD zum Absturz bringen kann!

SIEHE AUCH

PAD_Read

7.22 PDnet_ExtLifeList

AUFGABE

Liefert die erweiterte Lifeliste eines PDnet-Controllers.

PASCAL DEFINITION

```
function PDnet_ExtLifeList(
  PDnetId:            UINT8;
  var ExtLifeListTln: tExtLifeListTln
) : INT16;
```

C DEFINITION

```
INT16 PDnet_ExtLifeList(
  UINT8           PDnetId,
  tExtLifeListTln *ExtLifeListTln
);
```

EINGABEWERTE

- PDnetId
  In PDnetId wird die PDnet Stationsadresse des abzufragenden PADs angegeben.

AUSGABEWERTE

- ExtLifeListTln
  Über ExtLifeListTln wird das Array tExtLifeListTln mit den Daten der erweiterten Lifeliste zurückgegeben.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Die erweiterte Lifeliste wird ab der Firmwareversion 2.x unterstützt. In der erweiterten Lifeliste stehen für jeden PAD der online ist zusätzliche Daten über seine parametrierten Treiber.

SIEHE AUCH

tExtLifeListTln

7.23 PDnet_ExtOnLineStatus

AUFGABE

Ermittelt, über welche LAN-Kanäle ein PAD im PDnet erreichbar ist.

PASCAL DEFINITION

```
function PDnet_ExtOnLineStatus(
  PDnetId:  UINT8;
  var aBus: UINT8;
  var bBus: UINT8
) : INT16;
```

C DEFINITION

```
INT16 PDnet_ExtOnLineStatus(
  UINT8  PDnetId,
  UINT8 *aBus,
  UINT8 *bBus
);
```

EINGABEWERTE

- PDnetId
  In PDnetId wird die PDnet Stationsadresse des abzufragenden PADs angegeben.

AUSGABEWERTE

- aBus
  Über aBus wird ein Wert ungleich 0 zurückgegeben, wenn der Teilnehmer über den Kanal LAN-A erreichbar ist. Andernfalls ist aBus 0.
- bBus
  Über bBus wird ein Wert ungleich 0 zurückgegeben, wenn der Teilnehmer über den Kanal LAN-B erreichbar ist. Andernfalls ist bBus 0.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Diese Funktion ist für redundante PADs relevant, da sie die Verfügbarkeit eines PAD über die beiden LAN-Kanäle getrennt auswertet. Die Funktion prüft anhand der PDnet Lifeliste, ob der PAD über die LAN-Kanäle erreichbar ist.

PASCAL BEISPIEL

```
procedure TeilnehmerStatus(aPDnetId: Byte)
  function Status(s: UINT8) : string
  begin
    if s > 0 then
      status:='Ok'
    else
      status:='Offline';
  end;

var
  aBus, bBus: UINT8;
begin
  write('Teilnehmer ', aPDnetId, '  ');
  if PDnet_ExtOnLineStatus(aPDnetId,aBus,bBus)
  = CPDNET_OK then begin
    writeln('Online');
    writeln('A-BUS ',Status(aBus));
    writeln('B-Bus ',Status(bBus));
  end else
    writeln('Offline');
end;
```

C BEISPIEL

```
char* Status(UINT16 s);
{
  if (s)
    return("Ok");
  else
    return("Offline");
}

void TeilnehmerStatus(UINT8 aPDnetId);
{
  UINT8 aBus, bBus;
  printf("Teilnehmer %d ", aPDnetId);
  if (PDnet_ExtOnLineStatus(PDnetId, &aBus,
      &bBus) ==CPDNET_OK)
  {
    printf("Online\n");
    printf("A-BUS %s\n", Status(aBus));
    printf("B-Bus %s\n", Status(bBus));
  }
  else
  {
    printf("Offline\n");
  }
}
```

SIEHE AUCH

PDnet_OnLineStatus

7.24 PDnet_FlagByte

AUFGABE

Liefert für einen PAD im PDnet das Flagbyte der Lifeliste.

PASCAL DEFINITION

```
function PDnet_FlagByte(
  PDnetId:      UINT8;
  var FlagByte: UINT8
) : INT16;
```

C DEFINITION

```
INT16 PDnet_FlagByte(
  UINT8  PDnetId,
  UINT8 *FlagByte
);
```

EINGABEWERTE

- PDnetId
  In PDnetId wird die PDnet Stationsadresse des abzufragenden PADs angegeben.

AUSGABEWERTE

- FlagByte
  Über FlagByte wird der Eintrag der PDnet Lifeliste der abgefragten Station zurückgegeben.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Diese Funktion gibt das interne Flagbyte aus der Lifeliste für die abgefragte Station zurück. Für Anwenderprogramme ist diese Funktion ohne Bedeutung, da die Funktionen PDnet_OnLineStatus und PDnet_ExtOnLineStatus den LAN-Status des Flagbyte liefern.

SIEHE AUCH

PDnet_GroupAdr, PDnet_OnLineStatus, PDnet_ExtOnLineStatus

7.25 PDnet_GroupAdr

AUFGABE

Liefert die Gruppenadresse eines PAD im PDnet.

PASCAL DEFINITION

```
function PDnet_GroupAdr(
  PDnetId:      UINT8;
  var GroupAdr: UINT8
) : INT16;
```

C DEFINITION

```
INT16 PDnet_GroupAdr(
  UINT8  PDnetId,
  UINT8 *GroupAdr
);
```

EINGABEWERTE

- PDnetId
  In PDnetId wird die PDnet Stationsadresse des abzufragenden PADs angegeben.

AUSGABEWERTE

- GroupAdr
  Über GroupAdr wird die Gruppenadresse der abgefragten Station zurückgegeben.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Die Funktion bestimmt die Gruppenadresse des Teilnehmers über die PDnet Lifeliste.

C BEISPIEL

```
UINT8 PDnetId;
UINT8 GoupAdr;

printf("Teilnehmer-ID: ");
scanf("%d", &PDnetId);
if (PDnet_GoupId(PDnetId,&GoupAdr) == CPDNET_OK)
{
  printf("Station %d hat die Gruppenadresse %d",
         PDnetId, GroupAdr);
}
```

SIEHE AUCH

PDnet_OnLineStatus, PDnet_ExtOnLineStatus

7.26 PDnet_GroupMembers

AUFGABE

Liefert eine Liste aller Online Stationen die zu einer bestimmten Gruppe gehören.

PASCAL DEFINITION

```
function PDnet_GroupMembers(
  aGroupId: UINT8;
  var StationIDs: TStationIDs
) : INT16;
```

C DEFINITION

```
INT16 PDnet_GroupMembers(
  UINT8   aGroupID,
  TStationIDs StationIDs
);
```

EINGABEWERTE

- aGroupID
  In aGroupID wird die abzufragende Gruppenadresse übergeben.

AUSGABEWERTE

- StationIDs
  Über StationIDs wird ein Array mit den Stationsadressen der Gruppe zurückgegeben. Hinter der letzten Station steht eine 0 als Endekennung. Der Speicher für das Array StationIDs muß vor dem Aufruf der Funktion angelegt werden.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Die Funktion liefert aus der PDnet Lifeliste alle Stationsadressen der PADs, die der Gruppe angehören und online sind.

PASCAL BEISPIEL

```
procedure GruppenUebersicht;
  procedure show( grp: byte; liste: TStationIDs);
  var loop: byte;
  begin
    loop := 0;
    write('PDnet-Gruppe: ',grp,' ');
    while liste[loop] <> 0 do
    begin
      write(byte(liste[loop]):3,' ');
      inc(loop);
    end;
    writeln;
end;

var
  liste: TStationIDs
  found: byte;
  loop: byte;
begin
  found := 0;
  for loop := 1 to $ff do begin
    if PDnet_GroupMembers(loop, liste)
      = CPDNET_OK then
    begin
      show(loop, liste);
      if liste[0] > 0 then
        inc(found);
    end;
  end;
  if found = 0 then
    writeln('keine PDnet-Teilnehmer vorhanden')
  else
    writeln(found,' PDnet-Gruppen');
end.
```

C BEISPIEL

```
TStationIDs StationIDs
UINT8 loop;
UINT8 group;

printf("Bitte gewuenschte Gruppe eingeben: ");
scanf("%d", &group);
if (PDnet_GroupMembers(group, StationIDs)
  == CPDNET_OK)
{
  printf("Zur Gruppe %d gehören die Teilnehmer:",
         group);
  for(loop = 0; list[loop] != 0; loop++)
  {
    if (loop > 0)
      printf(", ");
    printf("%d", list[loop]);
  }
  printf("\n");
}
```

SIEHE AUCH

PDnet_GroupAdr

7.27 PDnet_LifeListChanged

AUFGABE

Meldet Änderungen der PDnet Lifeliste.

PASCAL DEFINITION

```
function PDnet_LifeListChanged(
  var Flag: INT16
) : INT16;
```

C DEFINITION

```
INT16 PDnet_LifeListChanged(
  INT16 *Flag
);
```

AUSGABEWERTE

- Flag
  Über Flag wird ein Wert ungleich 0 zurückgegeben, wenn sich die Lifeliste verändert hat. Andernfalls ist Flag 0.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Die Funktion prüft, ob sich in der PDnet Lifeliste ein PAD angemeldet oder abgemeldet hat und meldet dann die Änderung der Lifeliste. Diese Funktion sollte zyklisch aufgerufen werden um auf ein An-/Abmelden einzelner Stationen entsprechend zu reagieren. Das Anwenderprogramm kann sich z.B. eine Kopie der Lifeliste merken und bei Änderungen der Lifeliste diese mit der Funktion PDnet_OnLineStatus oder PDnet_ExtOnLineStatus neu einlesen, um die an- bzw. abgemeldeten Stationen zu melden.

PASCAL BEISPIEL

```
var ILifeListe: array[1..$ff] of boolean;

procedure UeberpruefeLifeListe;
var
  loop:      byte;
  online:    UINT8;
  LLChanged: INT16;
begin
  if(PDnet_LifeListChanged(LLChanged)<>CPDNET_OK)
  or (LLChanged = 0) then
    exit;
  for loop := 1 to $ff do
  begin
    if PDnet_OnLineStatus(loop, online)
      = CPDNET_OK then
      if ILifeListe[loop] <> online then
      begin
        write('Teilnehmer ',loop);
        if online > 0 then
          writeln(' angemeldet')
        else
          writeln(' abgemeldet');
        ILifeListe[loop] := online;
      end;
  end;
end;

begin { Hauptprogramm }
  fillchar(ILifeListe,SizeOf(ILifeListe),#0);
  ...
  while not ...
  begin
    ...
    repeat
      UeberprufeLifeListe;
    until keypressed;
  end;
end.
```

C BEISPIEL

```
INT16 LifeListChanged = 0;
if (PDnet_LifeListChanged(&LifeListChanged) ==
    CPDNET_OK)
{
  if (LifeListChanged != 0)
    printf("Die Lifeliste hat sich geändert!\n");
}
```

SIEHE AUCH

PDnet_OnLineStatus, PDnet_ExtOnLineStatus

7.28 PDnet_OnLineNodeCount

AUFGABE

Ermittelt die Anzahl der PADs die zur Zeit online sind.

PASCAL DEFINITION

```
function PDnet_OnLineNodeCount(
  var Count: UINT8
) : INT16;
```

C DEFINITION

```
INT16 PDnet_OnLineNodeCount(
  UINT8 *Count
);
```

AUSGABEWERTE

- Count
  Über Count wird die Anzahl aller PADs zurückgeliefert die in der PDnet Lifeliste online sind.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Die Funktion zählt alle online Einträge der PDnet Lifeliste und liefert deren Anzahl.

PASCAL BEISPIEL

```
count: UINT8;

if (PDnet_OnLineNodeCount(count) = CPDNET_OK)
then begin
  writeln('Zur Zeit sind ', count,
          ' Teilnehmer Online.");
end;
```

C BEISPIEL

```
UINT8 count;

if (PDnet_OnLineNodeCount(&count) == CPDNET_OK)
{
  printf("Zur Zeit sind %d Teilnehmer Online.\n",
         count);
}
```

SIEHE AUCH

PDnet_StationOnLine, PDnet_ExtStationOnLine, PDnet_GroupAdr

7.29 PDnet_OnLineStatus

AUFGABE

Ermittelt, ob ein PAD im PDnet online ist.

PASCAL DEFINITION

```
function PDnet_OnLineStatus(
  PDnetId:    UINT8;
  var Online: UINT8
) : INT16;
```

C DEFINITION

```
INT16 PDnet_OnLineStatus(
  UINT8  PDnetId,
  UINT8 *Online
);
```

EINGABEWERTE

- PDnetId
  In PDnetId wird die PDnet Stationsadresse des abzufragenden PADs angegeben.

AUSGABEWERTE

- Online
  Über Online wird ein Wert ungleich 0 zurückgegeben, wenn der Teilnehmer online ist. Andernfalls ist Online 0.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CPDNET_OK zurückgegeben. Andernfalls wird ein CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Die Funktion prüft, anhand der PDnet Lifeliste, ob die Station online ist.

C BEISPIEL

```
INT16 x, y;
UINT8 online;
UINT8 StationID;

/* Stationsadresse des lokalen PAD */
if (PAD_GetStationID(StationID) != CPDNET_OK)
{
  return;
}
printf("    ");
for (x=0; x<0x10; x++)
  printf("%3d ", x);        /* Spaltennummern */
for (y=0; y<0x100; y+=0x10)
{
  printf("\n%3d ", y);      /* Zeilennummern */
  for (x=0; x<0x10; x++)
  {
    if (x+y == 0)           /* Teilnehmer-ID 0 */
      printf("    ");
    else
    {
      /* Eigene Teilnehmer-ID? */
      if (x+y == StationID)
        printf(" XX ");
      else
      {
        if (PDnet_OnLineStatus(x+y, &online)
            == CPDNET_OK)
        {
          if (online == 0)
            printf("--- ");
          else
            printf("%3d ", online);
        }
      }
    }
  }
}
```

SIEHE AUCH

PDnet_ExtOnLineStatus

7.30 VDM_Check_PV

AUFGABE

Überprüft, ob eine PV in einer VDM-Datenzelle existiert.

PASCAL DEFINITION

```
function VDM_Check_PV(
  aFlags:    UINT8;
  aDataType: UINT8;
  aAddrPos:  UINT16;
) : INT16;
```

C DEFINITION

```
INT16 VDM_Check_PV(
  UINT8  aFlags,
  UINT8  aDataType,
  UINT16 aAddrPos
);
```

EINGABEWERTE

- aFlags
  In aFlags wird CZF_TRANSMITCELL übergeben, wenn VDM Sendezellen überprüft werden sollen. Ist aFlags 0, werden VDM Empfangszellen überprüft.
- aDataType
  In aDataType wird der VDM-Datentyp des Prozeßwertes übergeben. Der Datentyp ist eine CDT_??? Konstante.
- aAddrPos
  In aAddrPos wird für den Prozeßwert die PV-Nummer im Endgerät übergeben.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CVDM_OK zurückgegeben. Andernfalls wird ein CVDM_??? bzw. CPDNET_??? Fehlercode geliefert. Z.B. CVDM_BAD_ADDRPOS, wenn die übergebene PV nicht in den VDM Sendezellen bzw. den VDM Empfangszellen gefunden wird.

BESCHREIBUNG

Die Funktion VDM_Check_PV überprüft, ob eine PV in einer der VDM-Datenzellen existiert. Mit dem Projektierungsprogramm NetPro werden die VDM-Datenzellen angelegt. Die erste Prozeßvariable jeder VDM-Datenzelle hat die parametrierte PV-Nummer. Bei den folgenden Prozeßvariablen wird die PV-Nummer erhöht, bis die parametrierte Anzahl erreicht ist.

ACHTUNG

Vor dem ersten Aufruf von PV_Check_PV muß VDM_Init aufgerufen werden!

SIEHE AUCH

VDM_Init_PV, VDM_Read_PV, VDM_Write_PV

7.31 VDM_Done_PV

AUFGABE

Gibt den von VDM_Init_PV angelegten Speicher frei.

PASCAL DEFINITION

```
function VDM_Done_PV : INT16;
```

C DEFINITION

```
INT16 VDM_Done_PV(
  void
);
```

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CVDM_OK zurückgegeben. Andernfalls wird ein CVDM_??? bzw. CPDNET_??? Fehlercode geliefert.

BESCHREIBUNG

Die Funktion gibt den internen Speicherbereich wieder frei, der von VDM_Init_PV angelegt wurde. Nach dem Aufruf von VDM_Done darf keine andere VDM_??? Funktion außer VDM_Init aufgerufen werden.

SIEHE AUCH

VDM_Init_PV, VDM_Read_PV, VDM_Write_PV,

VDM_Check_PV

7.32 VDM_GetNextVdmHeader

AUFGABE

Liefert den Kopf der nächsten VDM-Datenzelle.

PASCAL DEFINITION

```
function VDM_GetNextVdmHeader(
  var aVdmAddr:   UINT32;
  var aVdmHeader: tVdmHeader
) : INT16;
```

C DEFINITION

```
INT16 VDM_GetNextVdmHeader(
  UINT32     *aVdmAddr,
  tVdmHeader *aVdmHeader
);
```

EINGABEWERTE

- aVdmAddr
  Im Inhalt von aVdmAddr wird die PAD Speicheradresse der vorherigen VDM-Datenzelle übergeben. Ist aVdmAddr 0, wird der Kopf der ersten VDM-Datenzelle geliefert.

AUSGABEWERTE

- aVdmAddr
  Über aVdmAddr wird die PAD Speicheradresse der gelieferten VDM-Datenzelle zurückgegeben. aVdmAddr ist 0, wenn keine VDM-Datenzellen mehr folgen.
- aVdmHeader
  Über aVdmHeader wird der Kopf der VDM-Datenzelle zurückgegeben, wenn aVdmAddr ungleich 0 ist.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CVDM_OK zurückgegeben. Andernfalls wird ein CVDM_??? bzw. CPDNET_??? Fehlercode geliefert. Z.B. CVDM_NOT_FOUND, wenn keine VDM-Datenzellen parametriert wurden.

SIEHE AUCH

tVdmHeader

7.33 VDM_Init_PV

AUFGABE

Initialisiert die internen Strukturen für die VDM Funktionen.

PASCAL DEFINITION

```
function VDM_Init_PV : INT16;
```

C DEFINITION

```
INT16 VDM_Init_PV(void);
```

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CVDM_OK zurückgegeben. Andernfalls wird ein CVDM_??? bzw. CPDNET_??? Fehlercode geliefert. Z.B. CVDM_NOT_FOUND, wenn keine VDM-Datenzellen parametriert wurden.

BESCHREIBUNG

Die Funktion VDM_Init_PV legt einen internen Speicherbereich an, in dem die VDM-Datenzellenköpfe des lokalen PAD kopiert werden. Die Zugriffe auf den VDM werden dadurch beschleunigt, da die Bibliothek die Adreßlagen der VDM-Datenzellen kennt. Wird VDM_Init_PV mehrmals aufgerufen, legt die Funktion keinen neuen Speicherbereich an, sondern liest die aktuellen VDM-Datenzellenköpfe in den bereits angelegten Speicherbereich ein.

ACHTUNG

VDM_Init muß vor den anderen Funktionen VDM_??? aufgerufen werden, da die Funktionen auf Kopie der VDM-Datenzellenköpfe zugreifen!

Für Merker ist NetPro ab V2.60 erforderlich, da die Funktionen VDM_??? nur mit PV-Nummern arbeiten.

SIEHE AUCH

VDM_Done_PV, VDM_Read_PV, VDM_Write_PV,

VDM_Check_PV

7.34 VDM_Read_PV

AUFGABE

Liefert die nächste geänderte PV aus den VDM-Empfangszellen.

PASCAL DEFINITION

```
function VDM_Read_PV(
  var aPvVal:    PVVAL;
  var aDataType: UINT8;
  var aAddrPos:  UINT16;
  aDateFlag:     UINT8
) : INT16;
```

C DEFINITION

```
INT16 VDM_Read_PV(
  PVVAL  *aPvVal,
  UINT8  *aDataType,
  UINT16 *aAddrPos,
  UINT8   aDateFlag
);
```

EINGABEWERTE

- aDateFlag
  Steht in aDateFlag CFB_DATE_CHANGED, werden nur geänderte PV's gelesen. Steht in aDateFlag CFB_DATE_VALID, werden nur gültige PV's gelesen.

AUSGABEWERTE

- aPvVal
  Liefert eine Struktur mit dem gelesenen Prozeßwert. Je nach Datentyp muß das zugehörige Feld von aPvVal gelesen werden.
- aDataType
  Liefert den VDM-Datentyp der Prozeßwertes. Der Datentyp ist eine CDT_??? Konstante.
- aAddrPos
  Liefert für den Prozeßwert die PV-Nummer im Endgerät.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CVDM_OK zurückgegeben. Andernfalls wird ein CVDM_??? bzw. CPDNET_??? Fehlercode geliefert. Z.B. CVDM_NO_PV_CHANGED, wenn keine geänderte bzw. gültige PV gefunden wurde.

BESCHREIBUNG

Bei jedem Aufruf der Funktion VDM_Read_PV werden die VDM-Empfangszellen hinter dem zuletzt gelieferten Prozeßwert auf Änderungen überprüft. Wird ein geänderter Prozeßwert gefunden, liefert die Funktion diesen zurück.

ACHTUNG

Vor dem ersten Aufruf von VDM_Read_PV muß VDM_Init_PV aufgerufen werden!

Für Gleitkommazahlen verwendet die PAD-Interface Bibliothek das IEEE-Format. Auf Plattformen, die kein IEEE-Format unterstützen, können keine Gleitkommazahlen verwendet werden!

SIEHE AUCH

VDM_Init_PV, VDM_Write_PV, VDM_Check_PV

7.35 VDM_Write_PV

AUFGABE

Schreibt eine PV in alle zugehörigen VDM-Sendezellen.

PASCAL DEFINITION

```
function VDM_Write_PV(
  var aPvVal:    PVVAL;
  aDataType:     UINT8;
  aAddrPos:      UINT16;
  aDateFlag:     UINT8
) : INT16;
```

C DEFINITION

```
INT16 VDM_Write_PV(
  PVVAL *aPvVal,
  UINT8  aDataType,
  UINT16 aAddrPos,
  UINT8  aDateFlag
);
```

EINGABEWERTE

- aPvVal
  In aPvVal wird der zu schreibende Prozeßwert übergeben. Je nach Datentyp muß das zugehörige Feld von aPvVal beschrieben werden.
- aDataType
  In aDataType wird der VDM-Datentyp des Prozeßwertes übergeben. Der Datentyp ist eine CDT_??? Konstante.
- aAddrPos
  In aAddrPos wird für den Prozeßwert die PV-Nummer im Endgerät übergeben.
- aDateFlag
  Steht in aDateFlag CFB_DATE_CHANGED, wird der Prozeßwert zur Gegenstation gesendet. Steht in aDateFlag CFB_DATE_VALID, wird der Prozeßwert in der VDM-Datenzelle auf gültig gesetzt aber nicht gesendet.

RÜCKGABEWERT

- Bei erfolgreicher Ausführung wird CVDM_OK zurückgegeben. Andernfalls wird ein CVDM_??? bzw. CPDNET_??? Fehlercode geliefert. Z.B. CVDM_NO_SYNC, wenn der Prozeßwert nicht in die VDM-Datenzelle geschrieben werden konnte, weil sie in Bearbeitung vom VDM war.

BESCHREIBUNG

Die Funktion VDM_Write_PV schreibt einen Prozeßwert in alle VDM-Sendezellen mit dem übergebenen Datentyp und der übergebenen PV-Nummer. Der zu schreibende Prozeßwert muß in aPvVal in das Feld mit dem zugehörigen Datentyp eingetragen werden.

ACHTUNG

Vor dem ersten Aufruf von PV_Write_PV muß VDM_Init_PV aufgerufen werden!

Für Gleitkommazahlen verwendet die PAD-Interface Bibliothek das IEEE-Format. Auf Plattformen, die kein IEEE-Format unterstützen, können keine Gleitkommazahlen verwendet werden!

SIEHE AUCH

VDM_Init_PV, VDM_Read_PV, VDM_Check_PV

8 Beispielprogramm

Dieses Kapitel beschreibt das in der PAD-Interface Bibliothek enthaltende Beispielprogramm.

Auf der mitgelieferten Diskette befindet sich das Beispielprogramm "PADTEST" einschließlich Quelltext. Außerdem finden sie dort je nach dem verwendeten Compiler die zugehörigen Compiler Konfigurationsdatei, Projektdatei oder Makedatei.

ACHTUNG

Damit Sie das Beispielprogramm übersetzen können, müssen Sie bei Ihrem Compiler die Option Word-Alignment ausschalten, da im PAD die Daten auch auf ungeraden Adressen stehen können. Strukturen müssen auf 1 Byte-Alignment ausgerichtet werden (Borland: Datenausrichtung Byte; Microsoft: Struct Member Alignment: Byte; Watcom: Structure Alignment: 1 Byte Alignment)!

Je nach dem verwendeten Betriebssystem unterscheiden sich die Parameter von "PADTEST":

Plattform	Betriebssystem	Parameter	Beschreibung
IBM-PC	DOS (Real Mode)	/IO=xxx	I/O-Base-Adresse (Hex)
	DOS (Protected Mode) Windows 3.1	/MEM=xxxx	Segmentadresse. (Hex) von PAD-PC bzw. PAD-PG
	QNX 3.21 QNX 4.22	/H	Paramterhilfe
	OS/2 Warp 3/4	keine	Parameter werden aus der Datei CONFIG.SYS gelesen
	Windows 95 Windows NT 3.5x/4.00	keine	Parameter werden aus der Registrierdatenbank gelesen
SUN-IPC/IPX	SUN-OS 4.1.2	-Sx	SBus-Slotnummer vom PAD-SBus
		-H	Parameterhilfe
Motorola-8420	System V/m88k	-Bxxx	Blocknummer des PAD-VME (in 256kByte-Schritten)
		-H	Parameterhilfe

8.1 IBM-PC DOS (Real Mode)

Das Beispielprogramm PADTEST für DOS Real Mode ist für folgende Compiler verfügbar.

8.1.1 Borland Pascal 7.0

Wenn Sie alle mitgelieferten Dateien in ein Verzeichnis kopieren und aus diesem Verzeichnis Borland Pascal aufrufen, erzeugt das beiliegende Beispielprogramm "PADTEST.PAS" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Borland Pascal im Menü

Options|Directories

die Verzeichnisangaben für Ihr System anpassen.

Dem Testprogramm "PADTEST.EXE" können Sie als Parameter mit

```
/IO=xxx
```

die I/O-Portadresse und mit

```
/MEM=xxxx
```

die Segmentadresse des PAD-PC angeben. Mit dem Parameter

```
/H
```

erhalten Sie eine kurze Hilfe zu den Parametern.

Die Parameterangaben /MEM=.... und /IO=... müssen mit den Adressen übereinstimmen, die auf dem PAD-PC mit dem DIP-Schalter SW1 eingestellt sind.

8.1.2 Borland C++ 3.1

Wenn Sie alle mitgelieferten Dateien in ein Verzeichnis kopieren und aus diesem Verzeichnis Borland C++ aufrufen, erzeugt die beiliegende Projektdatei "PADTEST.PRJ" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Borland C++ im Menü

Options|Directories

die Verzeichnisangaben für Ihr System anpassen.

Dem Testprogramm "PADTEST.EXE" können Sie als Parameter mit

```
/IO=xxx
```

die I/O-Portadresse und mit

```
/MEM=xxxx
```

die Segmentadresse des PAD-PC angeben. Mit dem Parameter

```
/H
```

erhalten Sie eine kurze Hilfe zu den Parametern.

Die Parameterangaben /MEM=.... und /IO=... müssen mit den Adressen übereinstimmen, die auf dem PAD-PC mit dem DIP-Schalter SW1 eingestellt sind.

8.2 IBM-PC DOS (Protected Mode)

Damit Sie das Beispielprogramm starten können, muß sich die dynamische Linkbibliothek PADITF16.DLL im Applikationsverzeichnis oder Suchpfad befinden.

Das Beispielprogramm PADTEST für DOS Protected Mode ist für folgende Compiler verfügbar.

8.2.1 Borland Pascal 7.0

Wenn Sie alle mitgelieferten Dateien in ein Verzeichnis kopieren und aus diesem Verzeichnis Borland Pascal aufrufen, erzeugt das beiliegende Beispielprogramm "PADTEST.PAS" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Borland Pascal im Menü

Options|Directories

die Verzeichnisangaben für Ihr System anpassen.

Dem Testprogramm "PADTEST.EXE" können Sie als Parameter mit

```
/IO=xxx
```

die I/O-Portadresse und mit

```
/MEM=xxxx
```

die Segmentadresse des PAD-PC angeben. Mit dem Parameter

```
/H
```

erhalten Sie eine kurze Hilfe zu den Parametern.

Die Parameterangaben /MEM=.... und /IO=... müssen mit den Adressen übereinstimmen, die auf dem PAD-PC mit dem DIP-Schalter SW1 eingestellt sind.

HINWEIS

Das Beispielprogramm PADTEST.PAS läuft als Protected Mode Anwendung unter DOS.

8.3 IBM-PC Windows 3.1

Damit Sie das Beispielprogramm starten können, muß sich die dynamische Linkbibliothek PADITF16.DLL im Applikationsverzeichnis oder im Suchpfad befinden.

Das Beispielprogramm PADTEST für Windows 3.1 ist für folgende Compiler verfügbar.

8.3.1 Borland Pascal 7.0

Wenn Sie alle mitgelieferten Dateien in ein Verzeichnis kopieren und aus diesem Verzeichnis Borland Pascal aufrufen, erzeugt das beiliegende Beispielprogramm "PADTEST.PAS" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Borland Pascal im Menü

Options|Directories

die Verzeichnisangaben für Ihr System anpassen.

HINWEIS

Die Pascal-Unit WinCrt emuliert die Ein-/Ausgabefunktionen eines DOS-Fensters unter Windows 3.1. Das Beispielprogramm PADTEST.PAS läuft als Standard DOS-Anwendung unter Windows 3.1. Selbstverständlich können Sie die PAD-Interface Bibliothek auch in eine echte Windowsanwendung einbinden.

Damit das Programm "PADTEST.EXE" als Symbol in einem Fenster von Windows 3.1 erscheint, müssen Sie das gewünschte Fenster zuerst aktivieren. Danach wählen Sie aus dem Fenster Programmanager das Menü

Datei|Neu

In dem erscheinenden Fenster wählen Sie

Neu (*)Programm

und betätigen den Schalter “OK“.

In dem Fenster Programmeigenschaften geben Sie dann ein:

Beschreibung: Padtest

Befehlszeile: LAUFWERK:\PFAD\padtest.exe /MEM=D000 /IO=300

Arbeitsverzeichnis: LAUFWERK:\PFAD

und betätigen den Schalter “OK“.

Für LAUFWERK müssen Sie den Buchstaben des Laufwerks und für PFAD den Verzeichnispfad angeben, an der das Programm "PADTEST.EXE" steht.

Dem Testprogramm "PADTEST.EXE" können Sie als Parameter mit

```
/IO=xxx
```

die I/O-Portadresse und mit

```
/MEM=xxxx
```

die Segmentadresse des PAD-PC angeben. Mit dem Parameter

```
/H
```

erhalten Sie eine kurze Hilfe zu den Parametern.

Die Parameterangaben /MEM=.... und /IO=... müssen mit den Adressen übereinstimmen, die auf dem PAD-PC mit dem DIP-Schalter SW1 eingestellt sind.

8.3.2 Borland C++ 3.1

Wenn Sie alle mitgelieferten Dateien in ein Verzeichnis kopieren und aus diesem Verzeichnis Borland C++ aufrufen, erzeugt die beiliegende Projektdatei "PADTEST.PRJ" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Borland C++ im Menü

Options|Directories

die Verzeichnisangaben für Ihr System anpassen.

HINWEIS

Wenn die Funktion WinMain nicht existiert, nimmt Borland C++ an, daß es sich bei Ihrem Programm um eine DOS-Anwendung handelt. Borland C++ linkt dann automatisch die EasyWin-Bibliothek hinzu, die für die Ein-/Ausgabefunktionen ein DOS-Fenster unter Windows 3.1 emuliert. Das Beispielprogramm PADTEST.C läuft als Standard DOS-Anwendung unter Windows 3.1. Selbstverständlich können Sie die PAD-Interface Bibliothek auch in eine echte Windowsanwendung einbinden.

Damit das Programm "PADTEST.EXE" als Symbol in einem Fenster von Windows 3.1 erscheint, müssen Sie das gewünschte Fenster zuerst aktivieren. Danach wählen Sie aus dem Fenster Programmanager das Menü

Datei|Neu

In dem erscheinenden Fenster wählen Sie

Neu (*)Programm

und betätigen den Schalter “OK“.

In dem Fenster Programmeigenschaften geben Sie dann ein:

Beschreibung: Padtest

Befehlszeile: LAUFWERK:\PFAD\padtest.exe /MEM=D000 /IO=300

Arbeitsverzeichnis: LAUFWERK:\PFAD

und betätigen den Schalter “OK“.

Für LAUFWERK müssen Sie den Buchstaben des Laufwerks und für PFAD den Verzeichnispfad angeben, an der das Programm "PADTEST.EXE" steht.

Dem Testprogramm "PADTEST.EXE" können Sie als Parameter mit

```
/IO=xxx
```

die IO-Base-Adresse und mit

```
/MEM=xxxx
```

die Segmentadresse des PAD-PC angeben. Mit dem Parameter

```
/H
```

erhalten Sie eine kurze Hilfe zu den Parametern.

Die Parameterangaben /MEM=.... und /IO=... müssen mit den Adressen übereinstimmen, die auf dem PAD-PC mit dem DIP-Schalter SW1 eingestellt sind.

8.4 IBM-PC Windows 95

Bevor Sie das Beispielprogramm starten können, müssen Sie unter Windows 95 erst den Gerätetreiber PADITF32.VXD installieren. Nähere Hinweise zur Installation des Gerätetreibers finden Sie im Kapitel „Installation Treiber“ bzw. in der Datei README95.TXT auf der Lieferdiskette.

Außerdem muß sich die dynamische Linkbibliothek PADITF32.DLL im Applikationsverzeichnis oder im Suchpfad befinden.

Das Beispielprogramm PADTEST für Windows 95 ist für folgende Compiler verfügbar.

8.4.1 Borland Delphi 2.0

Wenn Sie alle mitgelieferten Dateien aus den Verzeichnissen SRC und DELPHI20 in ein Verzeichnis kopieren und aus diesem Verzeichnis Borland Delphi 2.0 aufrufen, erzeugt das beiliegende Beispielprogramm "PADTEST.DPR" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Borland Delphi 2.0 im Menü

Projekt|Optionen|Verzeichnisse

die Verzeichnisangaben für Ihr System anpassen.

HINWEIS

Das Beispielprogramm PADTEST.DPR wurde für die WIN32-Konsole erzeugt, damit es im Textmode in der Eingabeaufforderung von Windows 95 läuft. Selbstverständlich können Sie die PAD-Interface Bibliothek auch in eine echte Windowsanwendung einbinden.

Starten Sie das Programm "PADTEST.EXE" über den Explorer oder die Eingabeaufforderung von Windows 95. Das Programm benötigt keine Parameter, da die vom PAD verwendeten Ressourcen aus der Registrierdatenbank von Windows 95 gelesen werden.

ACHTUNG

Die Bibliothek greift über die dynamische Linkbibliothek PADITF32.DLL auf den Gerätetreiber PADITF32.VXD zu. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

8.4.2 Borland C++ 4.5

Wenn Sie alle Dateien aus den Verzeichnissen SRC und W95BC45 in ein Verzeichnis kopieren und aus diesem Verzeichnis Borland C++ aufrufen, erzeugt die beiliegende Projektdatei "PADTEST.IDE" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Borland C++ im Menü

Optionen|Projekt|Verzeichnisse

die Verzeichnisangaben für Ihr System anpassen.

HINWEIS

Das Beispielprogramm PADTEST.C wurde für die WIN32-Konsole erzeugt, damit es im Textmode in der Eingabeaufforderung von Windows 95 läuft. Selbstverständlich können Sie die PAD-Interface Bibliothek auch in eine echte Windowsanwendung einbinden.

Starten Sie das Programm "PADTEST.EXE" über den Explorer oder die Eingabeaufforderung von Windows 95. Das Programm benötigt keine Parameter, da die vom PAD verwendeten Ressourcen aus der Registrierdatenbank von Windows 95 gelesen werden.

ACHTUNG

Die Bibliothek greift über die dynamische Linkbibliothek PADITF32.DLL auf den Gerätetreiber PADITF32.VXD zu. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

8.4.3 Microsoft Visual C++ 2.0

Wenn Sie alle Dateien aus den Verzeichnissen SRC und W95MSVC2 in ein Verzeichnis kopieren und aus diesem Verzeichnis Visual C++ 2.0 aufrufen, erzeugt die beiliegende Projektdatei "PADTEST.VCP" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Visual C++ im Menü

Project|Files|Files in Group

die Verzeichnisangaben für Ihr System anpassen.

HINWEIS

Das Beispielprogramm PADTEST.C wurde für die WIN32-Konsole erzeugt, damit es im Textmode in der Eingabeaufforderung von Windows 95 läuft. Selbstverständlich können Sie die PAD-Interface Bibliothek auch in eine echte Windowsanwendung einbinden.

Starten Sie das Programm "PADTEST.EXE" über den Explorer oder die Eingabeaufforderung von Windows 95. Das Programm benötigt keine Parameter, da die vom PAD verwendeten Ressourcen aus der Registrierdatenbank von Windows 95 gelesen werden.

ACHTUNG

Die Bibliothek greift über die dynamische Linkbibliothek PADITF32.DLL auf den Gerätetreiber PADITF32.VXD zu. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei

Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

8.4.4 Microsoft Visual C++ 4.0

Wenn Sie alle Dateien aus den Verzeichnissen SRC und W95MSVC4 in ein Verzeichnis kopieren und aus diesem Verzeichnis Visual C++ 4.0 aufrufen, erzeugt die beiliegende Projektdatei "PADTEST.MDP" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Visual C++4.0 im Register

FileView

die Projektdateien löschen und im Menü

Insert|Files into Project

die neuen Projektdateien eintragen.

HINWEIS

Das Beispielprogramm PADTEST.C wurde für die WIN32-Konsole erzeugt, damit es im Textmode in der Eingabeaufforderung von Windows 95 läuft. Selbstverständlich können Sie die PAD-Interface Bibliothek auch in eine echte Windowsanwendung einbinden.

Starten Sie das Programm "PADTEST.EXE" über den Explorer oder die Eingabeaufforderung von Windows 95. Das Programm benötigt keine Parameter, da die vom PAD verwendeten Ressourcen aus der Registrierdatenbank von Windows 95 gelesen werden.

ACHTUNG

Die Bibliothek greift über die dynamische Linkbibliothek PADITF32.DLL auf den Gerätetreiber PADITF32.VXD zu. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

8.4.5 Watcom C++ 10.0

Wenn Sie alle Dateien aus den Verzeichnissen SRC und W95WAT10 in ein Verzeichnis kopieren und aus diesem Verzeichnis Watcom C++ aufrufen, erzeugt die beiliegende Projektdatei "PADTEST.WPJ" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Watcom C++ im Menü

Sources|New Source

die Verzeichnisangaben für Ihr System anpassen.

HINWEIS

Das Beispielprogramm PADTEST.C wurde für die WIN32-Konsole erzeugt, damit es im Textmode in der Eingabeaufforderung von Windows 95 läuft. Selbstverständlich können Sie die PAD-Interface Bibliothek auch in eine echte Windowsanwendung einbinden.

Starten Sie das Programm "PADTEST.EXE" über den Explorer oder die Eingabeaufforderung von Windows 95. Das Programm benötigt keine Parameter, da die vom PAD verwendeten Ressourcen aus der Registrierdatenbank von Windows 95 gelesen werden.

ACHTUNG

Die Bibliothek greift über die dynamische Linkbibliothek PADITF32.DLL auf den Gerätetreiber PADITF32.VXD zu. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

8.5 IBM-PC Windows NT 3.5x/4.00

Bevor Sie das Beispielprogramm starten können, müssen Sie unter Windows NT erst den Gerätetreiber PADITF32.SYS installieren. Nähere Hinweise zur Installation des Gerätetreibers finden Sie im Kapitel „Installation Treiber“ bzw. in den Dateien READNT35.TXT und READNT40.TXT auf der Lieferdiskette.

Außerdem muß sich die dynamische Linkbibliothek PADITF32.DLL im Applikationsverzeichnis oder im Suchpfad befinden.

Das Beispielprogramm PADTEST für Windows NT 3.5x/4.00 ist für folgende Compiler verfügbar.

8.5.1 Borland Delphi 2.0

Wenn Sie alle mitgelieferten Dateien aus den Verzeichnissen SRC und DELPHI20 in ein Verzeichnis kopieren und aus diesem Verzeichnis Borland Delphi 2.0 aufrufen, erzeugt das beiliegende Beispielprogramm "PADTEST.DPR" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Borland Delphi 2.0 im Menü

Projekt|Optionen|Verzeichnisse

die Verzeichnisangaben für Ihr System anpassen.

HINWEIS

Das Beispielprogramm PADTEST.DPR wurde für die WIN32-Konsole erzeugt, damit es im Textmode in der Eingabeaufforderung von Windows NT läuft. Selbstverständlich können Sie die PAD-Interface Bibliothek auch in eine echte Windowsanwendung einbinden.

Starten Sie das Programm "PADTEST.EXE" über den Dateimanager oder die Eingabeaufforderung von Windows NT. Das Programm benötigt keine Parameter, da die vom PAD verwendeten Ressourcen aus der Registrierdatenbank von Windows NT gelesen werden.

ACHTUNG

Die Bibliothek greift über die dynamische Linkbibliothek PADITF32.DLL auf den Gerätetreiber PADITF32.SYS zu. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

8.5.2 Borland C++ 4.5

Wenn Sie alle Dateien aus den Verzeichnissen SRC und WNTBC45 in ein Verzeichnis kopieren und aus diesem Verzeichnis Borland C++ aufrufen, erzeugt die beiliegende Projektdatei "PADTEST.IDE" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Borland C++ im Menü

Optionen|Projekt|Verzeichnisse

die Verzeichnisangaben für Ihr System anpassen.

HINWEIS

Das Beispielprogramm PADTEST.C wurde für die WIN32-Konsole erzeugt, damit es im Textmode in der Eingabeaufforderung von Windows NT läuft. Selbstverständlich können Sie die PAD-Interface Bibliothek auch in eine echte Windowsanwendung einbinden.

Starten Sie das Programm "PADTEST.EXE" über den Dateimanager oder die Eingabeaufforderung von Windows NT. Das Programm benötigt keine Parameter, da die vom PAD verwendeten Ressourcen aus der Registrierdatenbank von Windows NT gelesen werden.

ACHTUNG

Die Bibliothek greift über die dynamische Linkbibliothek PADITF32.DLL auf den Gerätetreiber PADITF32.SYS zu. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

8.5.3 Microsoft Visual C++ 2.0

Wenn Sie alle Dateien aus den Verzeichnissen SRC und WNTMSVC2 in ein Verzeichnis kopieren und aus diesem Verzeichnis Visual C++ 2.0 aufrufen, erzeugt die beiliegende Projektdatei "PADTEST.VCP" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Visual C++ im Menü

Project|Files|Files in Group

die Verzeichnisangaben für Ihr System anpassen.

HINWEIS

Das Beispielprogramm PADTEST.C wurde für die WIN32-Konsole erzeugt, damit es im Textmode in der Eingabeaufforderung von Windows NT läuft. Selbstverständlich können Sie die PAD-Interface Bibliothek auch in eine echte Windowsanwendung einbinden.

Starten Sie das Programm "PADTEST.EXE" über den Dateimanager oder die Eingabeaufforderung von Windows NT. Das Programm benötigt keine Parameter, da die vom PAD verwendeten Ressourcen aus der Registrierdatenbank von Windows NT gelesen werden.

ACHTUNG

Die Bibliothek greift über die dynamische Linkbibliothek PADITF32.DLL auf den Gerätetreiber PADITF32.SYS zu. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

8.5.4 Microsoft Visual C++ 4.0

Wenn Sie alle Dateien aus den Verzeichnissen SRC und WNTMSVC4 in ein Verzeichnis kopieren und aus diesem Verzeichnis Visual C++ 4.0 aufrufen, erzeugt die beiliegende Projektdatei "PADTEST.MDP" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Visual C++4.0 im Register

FileView

die Projektdateien löschen und im Menü

Insert|Files into Project

die neuen Projektdateien eintragen.

HINWEIS

Das Beispielprogramm PADTEST.C wurde für die WIN32-Konsole erzeugt, damit es im Textmode in der Eingabeaufforderung von Windows NT läuft. Selbstverständlich können Sie die PAD-Interface Bibliothek auch in eine echte Windowsanwendung einbinden.

Starten Sie das Programm "PADTEST.EXE" über den Dateimanager oder die Eingabeaufforderung von Windows NT. Das Programm benötigt keine Parameter, da die vom PAD verwendeten Ressourcen aus der Registrierdatenbank von Windows NT gelesen werden.

ACHTUNG

Die Bibliothek greift über die dynamische Linkbibliothek PADITF32.DLL auf den Gerätetreiber PADITF32.SYS zu. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

8.5.5 Watcom C++ 10.0

Wenn Sie alle Dateien aus den Verzeichnissen SRC und WNTWAT10 in ein Verzeichnis kopieren und aus diesem Verzeichnis Watcom C++ aufrufen, erzeugt die beiliegende Projektdatei "PADTEST.WPJ" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Watcom C++ im Menü

Sources|New Source

die Verzeichnisangaben für Ihr System anpassen.

HINWEIS

Das Beispielprogramm PADTEST.C wurde für die WIN32-Konsole erzeugt, damit es im Textmode in der Eingabeaufforderung von Windows NT läuft. Selbstverständlich können Sie die PAD-Interface Bibliothek auch in eine echte Windowsanwendung einbinden.

Starten Sie das Programm "PADTEST.EXE" über den Dateimanager oder die Eingabeaufforderung von Windows NT. Das Programm benötigt keine Parameter, da die vom PAD verwendeten Ressourcen aus der Registrierdatenbank von Windows NT gelesen werden.

ACHTUNG

Die Bibliothek greift über die dynamische Linkbibliothek PADITF32.DLL auf den Gerätetreiber PADITF32.SYS zu. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

8.6 IBM-PC OS/2 Warp 3/4

Bevor Sie das Beispielprogramm starten können, müssen Sie unter OS/2 erst den Einheitentreiber APEX-PAD.SYS installieren. Nähere Hinweise zur Installation des Einheitentreibers finden Sie im Kapitel „Installation Treiber“ bzw. in der Datei READMEO2.TXT auf der Lieferdiskette.

Außerdem muß sich die dynamische Linkbibliothek PADITF32.DLL im Applikationsverzeichnis oder im Suchpfad befinden.

Das Beispielprogramm PADTEST für OS/2 Warp ist für folgende Compiler verfügbar.

8.6.1 Borland C++ 2.0 für OS/2

Wenn Sie alle Dateien aus den Verzeichnissen SRC und OS2BC20 in ein Verzeichnis kopieren und aus diesem Verzeichnis Borland C++ aufrufen, erzeugt die beiliegende Projektdatei "PADTEST.PRJ" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Borland C++ im Menü

Project|View settings|Directories

die Verzeichnisangaben für Ihr System anpassen.

HINWEIS

Das Beispielprogramm PADTEST.C wurde für die OS/2-Konsole erzeugt, damit es im Textmode in einem OS/2-Fenster läuft. Selbstverständlich können Sie die PAD-Interface Bibliothek auch in eine echte OS/2-Anwendung einbinden.

Starten Sie das Programm "PADTEST.EXE" über ein Symbolanzeigefenster oder ein OS/2-Fenster. Das Programm benötigt keine Parameter, da die vom PAD verwendeten Ressourcen aus der Datei CONFIG.SYS von OS/2 gelesen werden.

ACHTUNG

Die Bibliothek greift über die dynamische Linkbibliothek PADITF32.DLL auf den Einheitentreiber APEX-PAD.SYS zu. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

8.6.2 VisualAge C++ 3.0

Wenn Sie alle Dateien aus den Verzeichnissen SRC und OS2VAGE3 in ein Verzeichnis kopieren und aus diesem Verzeich-

nis VisualAge C++ aufrufen, erzeugt die beiliegende Projektdatei "PADTEST" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im VisualAge C++ im Menü

View|Settings|Location

die Verzeichnisangaben für Ihr System anpassen.

HINWEIS

Das Beispielprogramm PADTEST.C wurde für die OS/2-Konsole erzeugt, damit es im Textmode in einem OS/2-Fenster läuft. Selbstverständlich können Sie die PAD-Interface Bibliothek auch in eine echte OS/2-Anwendung einbinden.

Starten Sie das Programm "PADTEST.EXE" über ein Symbolanzeigefenster oder ein OS/2-Fenster. Das Programm benötigt keine Parameter, da die vom PAD verwendeten Ressourcen aus der Datei CONFIG.SYS von OS/2 gelesen werden.

ACHTUNG

Die Bibliothek greift über die dynamische Linkbibliothek PADITF32.DLL auf den Einheitentreiber APEX-PAD.SYS zu. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

8.6.3 Watcom C++ 10.0

Wenn Sie alle Dateien aus den Verzeichnissen SRC und OS2WAT10 in ein Verzeichnis kopieren und aus diesem Verzeichnis Watcom C++ aufrufen, erzeugt die beiliegende Projektdatei "PADTEST.WPJ" das Testprogramm "PADTEST.EXE".

Eventuell müssen Sie im Watcom C++ im Menü

Sources|New Source

die Verzeichnisangaben für Ihr System anpassen.

HINWEIS

Das Beispielprogramm PADTEST.C wurde für die OS/2-Konsole erzeugt, damit es im Textmode in einem OS/2-Fenster läuft. Selbstverständlich können Sie die PAD-Interface Bibliothek auch in eine echte OS/2-Anwendung einbinden.

Starten Sie das Programm "PADTEST.EXE" über ein Symbolanzeigefenster oder ein OS/2-Fenster. Das Programm benötigt keine Parameter, da die vom PAD verwendeten Ressourcen aus der Datei CONFIG.SYS von OS/2 gelesen werden.

ACHTUNG

Die Bibliothek greift über die dynamische Linkbibliothek PADITF32.DLL auf den Einheitentreiber APEX-PAD.SYS zu. Die PADITF32.DLL ist zwar reentrantfähig, wenn die DLL jedoch von mehreren Programmen gleichzeitig benutzt wird, dürfen keine zwei Programme auf den gleichen PAD-Speicher zugreifen, da sie sich sonst gegenseitig die Geändertflags löschen.

8.7 IBM-PC QNX 3.21 (Protected Mode)

Das Beispielprogramm PADTEST für QNX 3.21 (Protected Mode) ist für folgende Compiler verfügbar.

8.7.1 C86 für QNX

Die beiliegende Datei "makefile" erzeugt das Testprogramm "padtest".

Eventuell müssen Sie in der Datei "makefile" die Verzeichnisangaben für Ihr System anpassen.

Außerdem muß sich der C-Compiler C86 "cq" im Verzeichnis "/cmds" befinden und seine Include-Dateien im Verzeichnis "/cii/include".

Vor dem ersten Aufruf von C86 muß das Programm "dyna" in einem Hintergrundtask gestartet werden:

- `dyna &`

Dem Testprogramm "PADTEST.EXE" können Sie als Parameter mit

```
/IO=xxx
```

die I/O-Portadresse und mit

```
/MEM=xxxx
```

die Segmentadresse des PAD-PC angeben. Mit dem Parameter

```
/H
```

erhalten Sie eine kurze Hilfe zu den Parametern.

Die Parameterangaben /MEM=.... und /IO=... müssen mit den Adressen übereinstimmen, die auf dem PAD-PC mit dem DIP-Schalter SW1 eingestellt sind.

HINWEIS

Die Bibliotheken sind mit dem C-Compiler C86 für QNX ("cq") erzeugt, da dieser auch ein ANSI-C-Compiler ist. Folgende Parameter wurden zum Übersetzen der Bibliotheken und des Beispielprogramms "padtest" verwendet:

Parameter	Beschreibung
-AL	Speichermodell "Large"
-G2	Prozessor 80286 und höher
-Za	ANSI-C

Sollte die CPU Ihres verwendeten Computers keinen mathematischen Coprozessor besitzen, müssen Sie vor dem ersten Aufruf von dem C-Compiler C86 bzw. dem Beispielprogramm "padtest" den Coprozessor Emulator von C86 aufrufen:

- `cii_emul_8087 &`

8.8 IBM-PC QNX 4.22 (Protected Mode)

Das Beispielprogramm PADTEST für QNX 4.22 (Protected Mode) ist für folgende Compiler verfügbar.

8.8.1 Watcom C 9.5

Die beiliegende Makedatei "makefile" erzeugt das Testprogramm "padtest" mit dem Make-Utility:

- `# make`

Der C-Compiler muß mit dem Parameter -T1 aufgerufen werden, damit das Beispielprogramm auf die I/O-Ports zugreifen kann.

Evt. müssen Sie die Dateirechte des Beispielprogramms noch auf „ausführbar“ setzen:

- `# chmod +x padtest`

Sie starten das Beispielprogramm mit dem Befehl:

- `# ./padtest`

Dem Testprogramm "padtest" können Sie als Parameter mit

`/IO=xxx`

die I/O-Portadresse und mit

`/MEM=xxxx`

die Segmentadresse des PAD-PC angeben. Mit dem Parameter

`/H`

erhalten Sie eine kurze Hilfe zu den Parametern.

Die Parameterangaben /MEM=.... und /IO=... müssen mit den Adressen übereinstimmen, die auf dem PAD-PC mit dem DIP-Schalter SW1 eingestellt sind.

8.9 DEC AlphaStation OpenVMS/AXP 6.2/7.0

Das Beispielprogramm PADTEST für DEC AlphaStation OpenVMS/AXP ist für folgende Compiler verfügbar.

8.9.1 DEC C V5.2

Die Lieferdiskette besitzt das OpenVMS Dateiformat. Um sie auf der DEC AlphaStation zu lesen müssen Sie die Diskette mounten. Dieses geschieht in einem DECterm Fenster mit dem Befehl:

- `$ mount dva0: paditf`

Nach dem Mounten der Diskette können Sie die PAD-Interface Bibliothek mit dem Beispielprogramm aus dem Archiv PADIT017.A in das Verzeichnis SYS$SYSROOT:[SYSMGR.PADITF] installieren (siehe Installation Bibliothek).

Durch Aufruf der beiliegenden Kommandodatei "makefile.com" erzeigt das Testprogramm "padtest.exe":

- `$ @makefile`

Die Kommandodatei “makefile.com“ compiliert das Beispielprogramm:

```
cc padtest.c
```

Anschließend werden alle Dateien gelinkt:

```
link padtest.obj, paditf.obj, padlib.obj,
padvdm.obj, sys$library:padlib.olb/LIBRARY
```

Wenn Sie das Programm “padtest.exe“ mit Parametern aufrufen wollen, müssen Sie ein Symbol definieren:

- `$ padtest :== run SYS$SYSROOT:[SYSMGR.PADITF] padtest.exe`

Dem Symbol "padtest" können Sie als Parameter den Device Namen des Gerätetreibers übergeben:

- `$ padtest jda0:`

Fehlt der Parameter, wird das Gerät “jda0:“ verwendet.

HINWEIS

Die Bibliotheken sind mit dem ANSI-C-Compiler DEC C V5.2 ("cc") erzeugt. Da der Compiler nicht das IEEE-Format für Realvariablen besitzt, können keine Gleitpunktzahlen als Prozeßwert verwendet werden.

8.10 SUN IPC/IPX (SUN-OS 4.1.2)

Das Beispielprogramm PADTEST für SUN IPC/IPX (SUN-OS 4.1.2) ist für folgende Compiler verfügbar.

8.10.1 GNU-C

Die Lieferdiskette besitzt das PC-Dateiformat. Um sie auf der SUN-Workstation zu lesen müssen Sie das PC-FileSystem mounten. Dieses geschieht mit den Befehlen:

- `su`
(Superuser werden, evt. mit Paßwort)
- `mount -t pcfs /dev/fd0c /floppy`
(Diskette mit Name /floppy mounten)
- `cp -p /floppy`
(Alle Dateien ins aktuelle Verzeichnis kopieren)
- `eject`
(Diskette auswerfen)
- `exit`
(Superuser-Mode verlassen)

Die beiliegende Datei "makefile" erzeugt das Testprogramm "padtest". Eventuell müssen Sie in der Datei "makefile" die Verzeichnisangaben für Ihr System anpassen. Außerdem muß der Pfad auf den GNU-C-Compiler "gcc" gesetzt sein.

Dem Testprogramm "padtest" können Sie als Parameter mit

```
-Sx
```

die SBus-Slotnummer des PAD-SBus angeben. Mit dem Parameter

```
-H
```

erhalten Sie eine kurze Hilfe zu den Parametern.

Der Parameter -Sx muß mit der SBus-Slotnummer übereinstimmen, auf dem der PAD-SBus steckt. Außerdem muß auf Ihrem System die zugehörige Datei

```
/dev/sbus*
```

existieren und die Zugriffsrechte gesetzt sein.

HINWEIS

Die Bibliotheken sind mit dem GNU-C-Compiler 2.2 ("gcc") erzeugt, da dieser ein ANSI-C-Compiler ist und er auf verschiedenen Plattformen erhältlich ist. Der Compiler "cc" verarbeitet kein ANSI-C und besitzt nicht das IEEE-Format für Realvariablen.

Damit GNU-C die richtigen Include-Verzeichnisse von UNIX verwendet, müssen diese als Parameter angegeben werden:

```
-I/usr/include
```

Eventuell müssen Sie das Verzeichnis angeben, in dem die Bibliotheken und Datendateien des GNU-C stehen. Zum Beispiel:

```
-B/usr/local/lib/gcc-lib/sparc-sun-
sunos4.1/2.2.2/
```

Verwendete Quellen werden angegeben mit:

```
-o EXE_PROGRAMM_NAME SRC_PROGRAMM_NAME.c paditf.o
padlib.o
```

8.11 SUN Ultra 5 (Solaris 2.6)

Das Beispielprogramm PADTEST für SUN Ultra 5 (Solaris 2.6) ist für folgende Compiler verfügbar.

8.11.1 SunPro C 4.2

Die Lieferdiskette besitzt das PC-Dateiformat. Um sie auf der SUN-Workstation zu lesen müssen Sie das PC-FileSystem mounten. Dieses geschieht mit den Befehlen:

- `su`
(Superuser werden, evt. mit Paßwort)
- `mount -t pcfs /dev/fd0c /floppy`
(Diskette mit Name /floppy mounten)
- `cp -p /floppy`
(Alle Dateien ins aktuelle Verzeichnis kopieren)
- `eject`
(Diskette auswerfen)
- `exit`
(Superuser-Mode verlassen)

Die beiliegende Datei "makefile" erzeugt das Testprogramm "padtest". Eventuell müssen Sie in der Datei "makefile" die Verzeichnisangaben für Ihr System anpassen. Außerdem muß der Pfad auf den SunPro C-Compiler "cc" gesetzt sein.

Dem Testprogramm "padtest" können Sie als Parameter mit

```
-Px
```

die Gerätenummer des PAD-PCI angeben. Mit dem Parameter

```
-H
```

erhalten Sie eine kurze Hilfe zu den Parametern.

Der Parameter -Px muß mit der Gerätenummer im Gerätenamen „/dev/pdnetX“ übereinstimmen der zum PAD-PCI gehört. Außerdem muß auf Ihrem System der Gerätetreiber pdnet installiert sein.

HINWEIS

Damit SunPro C die richtigen Include-Verzeichnisse von UNIX verwendet, müssen diese als Parameter angegeben werden:

```
-I/usr/include
```

8.12 Motorola-8420 (System V/m88k)

Das Beispielprogramm PADTEST für Motorola-8420 (System V/m88k) ist für folgende Compiler verfügbar.

8.12.1 GNU-C

Die beiliegende Datei "makefile" erzeugt das Testprogramm "padtest".

Eventuell müssen Sie in der Datei "makefile" die Verzeichnisangaben für Ihr System anpassen. Außerdem muß der Pfad auf den GNU-C-Compiler "gcc" gesetzt sein.

Dem Testprogramm "padtest" können Sie als Parameter mit

```
-Bxxx
```

die Blocknummer des PAD-VME angeben. Mit dem Parameter

```
-H
```

erhalten Sie eine kurze Hilfe zu den Parametern.

Aus der Blocknummer des Parameters -Bxxx multipliziert mit 256 kByte ergibt sich die Adresse ab der das Speicherfenster des PAD-VME eingeblendet wird. Wenn Ihr System z. B. die ersten 32 MByte des Speichers belegt und der PAD-VME ab Adresse 32 MByte liegen soll ergibt sich eine Blocknummer von 32 MByte / 256 kByte = 128

Der Parameter wäre dann:

```
-B128
```

Beachten Sie auch die Hinweise zum Shared-Memory des VME-Bus im Kapitel Installation Hardware.

HINWEIS

Die Bibliotheken sind mit dem GNU-C-Compiler 2.3.1 ("gcc") erzeugt, da dieser ein ANSI-C-Compiler ist und er auf verschiedenen

Plattformen erhältlich ist. Der Compiler "cc" verarbeitet kein ANSI-C und besitzt nicht das IEEE-Format für Realvariablen.

Damit GNU-C die richtigen Include-Verzeichnisse von UNIX verwendet, müssen diese als Parameter angegeben werden:

```
-I/usr/include
```

Eventuell müssen Sie das Verzeichnis angeben, in dem die Bibliotheken und Datendateien des GNU-C stehen. Zum Beispiel:

```
-B/usr/local/lib/gcc-lib/m88k-sysv3/2.3.1/
```

Verwendete Quellen werden angegeben mit:

```
-o EXE_PROGRAMM_NAME SRC_PROGRAMM_NAME.c paditf.o
padlib.o
```

9 Programmierung

Bei der Programmierung von Treibern für das PDnet sollten Sie folgende Kapitel beachten.

9.1 Projektierungsprogramm NetPro

Mit dem Projektierungsprogramm NetPro werden die PDnet-Controller projektiert und die VDM-Datenzellen angelegt.

9.1.1 VDM-Datenzellen

Jede VDM-Datenzelle ist entweder eine Sendezelle oder eine Empfangszelle. Eine VDM-Datenzelle enthält mehrere Prozeßwerte mit gleichem Datentyp. In einer SPS werden die Prozeßwerte direkt in den SPS-Speicher geschrieben bzw. von ihm gelesen. Auf PCs und Workstations verarbeitet die PAD-Interface Bibliothek die Prozeßwerte der Prozeßvariablen (PV) über PV-Nummern. Jede PV-Nummer darf in den VDM-Empfangszellen nur einmal vorkommen. Ebenso darf jede PV-Nummer in den VDM-Sendezellen nur einmal vorkommen.

Bei der Projektierung von VDM-Datenzellen zwischen einer SPS und einem Leitsystem ist die Stellung des Parameters „Alles senden“ zu beachten:

VDM-Sendezelle	VDM-Empfangszelle	Alles senden	Beschreibung
SPS	Leitsystem	EIN	Beim Anmelden eines PDnet-Controllers werden alle Prozeßwerte gesendet. Ändert sich ein Prozeßwert der VDM-Datenzelle, werden alle Prozeßwerte der VDM-Datenzelle übertragen.
Leitsystem	SPS	AUS	Beim Anmelden eines PDnet-Controllers wird noch kein Prozeßwert gesendet. Ändert sich ein Prozeßwert der VDM-Datenzelle, wird nur der geänderte Prozeßwert übertragen.

9.1.2 PV Symbolnamen

Wenn Ihr Leitsystem mit Symbolnamen auf die Prozeßwerte zugreift, sollten Sie jeder PV-Nummer einen Symbolnamen zuordnen. Im Projektierungsprogramm NetPro kann für jede PV-Nummer ein Symbolname eingeben werden. Die Symbollisten können in verschiedene Formate exportiert und importiert werden, z.B. in ASCII- oder dBase-Dateien. Fragen Sie bei der APEX GmbH nach der Implementierung neuer Formate.

9.2 Lokalen PAD verwalten

Beim Programmstart muß die PAD-Interface Bibliothek mit der Funktion PAD_Init initialisiert werden, bevor auf andere API Funktionen der Bibliothek zugegriffen werden kann.

Das Anwenderprogramm muß in regelmäßigen Abständen eine der API-Funktionen aufrufen, um das PAD-Interface am Leben zu erhalten. Die Zykluszeit wird im Parameter PcCheckTmrVal an PAD_Init übergeben. Greift ein Anwenderprogramm nicht ständig auf den PAD zu, muß es zyklisch die Funktion PAD_LifeCheck aufrufen, um das PAD-Interface am Leben zu erhalten.

Liefert eine Funktion CPDNET_RESTART_RUNNING sollte PAD_CheckRestart zyklisch aufgerufen werden, bis die Funktion CPDNET_RESTART_END meldet. Erst danach ist ein Zugriff auf den PAD wieder möglich.

Beim Beenden des Programms muß die Funktion PAD_Done aufgerufen werden, um die Verbindung zum PAD zu trennen.

9.3 Lifeliste überwachen

Das Anwenderprogramm kann das an- und abmelden anderer PDnet-Controller überwachen, in dem beim Programmstart die Lifeliste mit der Funktion PDnet_OnLineStatus bzw. PDnet_ExtOnLineStatus in einen Pufferspeicher kopiert wird. Wenn der zyklische Aufruf der Funktion PDnet_LifeListChanged eine Änderung der PDnet Lifeliste meldet, kann die Lifeliste neu eingelesen werden und die Änderungen mit dem Pufferspeicher verglichen werden.

Änderungen der erweiterten Lifeliste werden von der Funktion PDnet_LifeListChanged nicht gemeldet. Deshalb muß eine Änderungsüberwachung der erweiterten Lifeliste zyklisch (z.B. jede Sekunde) die Funktion PDnet_ExtLifeList für die Stationen aufrufen, die online sind. Die Änderungen der erweiterten Lifeliste können ebenfalls mit einem Pufferspeicher verglichen werden, der beim Programmstart mit einer Kopie der erweiterten Lifeliste gefüllt wurde.

9.4 Telegramme senden und empfangen

Zum Senden von Telegrammen dient die Funktion PAD_SendTelegram. Der Empfang von Telegrammen erfolgt mit der Funktion PAD_ReceiveTelegram. Die Funktionen PAD_TxBufferEmpty und PAD_RxBufferEmpty sind nur zur Kompatibilität zu älteren Bibliotheksversionen noch enthalten. Sie sind eigentlich überflüssig, da die Funktionen PAD_SendTelegram und PAD_ReceiveTelegram ebenfalls melden, ob der Telegrammpuffer voll bzw. leer ist.

ACHTUNG

In einem Multitasking Betriebsystem darf nur ein Task die Funktion PAD_ReceiveTelegram aufrufen, da die Funktion nach dem Auslesen des Telegrammpuffers diesen als leer markiert. Wenn mehrere Task die Funktion PAD_ReceiveTelegram aufrufen, wird ein empfangenes Telegramm nur an einen der Tasks übermittelt.

9.5 Prozeßwerte senden und empfangen

Die Funktionen VDM_??? greifen auf die VDM-Datenzellen des lokalen PAD zu. Die Funktion VDM_Init muß aufgerufen werden, bevor auf andere VDM_??? Funktion zugegriffen werden kann.

Nach VDM_Init sollten einmalig alle gültigen Prozeßwerte eingelesen werden. Dazu wird die Funktion PAD_Read_PV mit dem Parameter CFB_DATE_VALID aufgerufen. Die erste gelieferte PV-Nummer sollten Sie sich merken und PAD_Read_PV mit dem Parameter CFB_DATE_VALID so lange aufrufen, bis die erste PV-Nummer wieder erreicht wurde.

Nach dem Einlesen der gültigen Prozeßwerte sollte das Anwenderprogramm nur noch geänderte Prozeßwerte verarbeiten, in dem es die Funktion PAD_Read_PV mit dem Parameter CFB_DATE_CHANGED zyklisch aufruft.

Das Senden von Prozeßwerten übernimmt die Funktion VDM_Write_PV.

Beim Beenden des Programms muß die Funktion VDM_Done aufgerufen werden, um den von VDM_Init angelegten Speicher freizugeben.

ACHTUNG

In einem Multitasking Betriebsystem darf nur ein Task die Funktion PAD_Read_PV aufrufen, da die Funktion nach dem Auslesen der VDM-Datenzelle das Änderungsbit des gelesenen Prozeßwertes löscht. Wenn mehrere Task die Funktion PAD_Read_PV aufrufen, wird ein geänderter Prozeßwert nur an einen der Tasks übermittelt.

Falls Sie ihr Anwenderprogramm in mehrere Task aufteilen wollen, sollte ein Task die Lifeliste überwachen, ein Task Telegramme senden, ein Task Telegramme empfangen, ein Task Prozeßwerte senden und ein Task Prozeßwerte empfangen.

